NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

A SPATIOTEMPORAL CLUSTERING APPROACH TO MARITIME DOMAIN AWARENESS

by

Kristofer A. Tester

September 2013

Thesis Advisor:	Jim Scrofani
Thesis Co-Advisor:	Murali Tummala

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704–0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202–4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704–0188) Washington, DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2013	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE A SPATIOTEMPORAL CLUSTERING APPROACH TO MARITIME DOMAIN AWARENESS			**5. FUNDING NUMBERS**
6. AUTHOR(S) Kristofer A. Tester			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943–5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

Spatiotemporal clustering is the process of grouping objects based on both their spatial and temporal similarity. This approach is useful when considering the distance between objects and how that distance changes through time. Spatiotemporal clustering analysis is applied to the maritime domain in this thesis, specifically to a defined area of water, during a period of time, in order to gain behavioral knowledge of vessel interactions and provide the opportunity to screen such interactions for further investigation. The proposed spatiotemporal clustering algorithm spatially clusters vessels in the water space using k-means clustering analysis, kinematically refines the clusters based on the similarity of vessel headings, speeds and the distance between them, and temporally analyzes the continuity of membership of the kinematic clusters through time to determine which clusters are moving. The algorithm is implemented in the MATLAB programming environment, verified with a synthetic data scenario, and validated with two real-world datasets.

14. SUBJECT TERMS Spatiotemporal Clustering; Maritime Domain Awareness			**15. NUMBER OF PAGES** 125
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540–01–280–5500 Standard Form 298 (Rev. 2–89) Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

A SPATIOTEMPORAL CLUSTERING APPROACH TO MARITIME DOMAIN AWARENESS

Kristofer A. Tester
Lieutenant, United States Navy
B.S., United States Naval Academy, 2006

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2013

Author:	Kristofer A. Tester
Approved by:	Jim Scrofani Thesis Advisor
	Murali Tummala Thesis Co-Advisor
	David Garren Second Reader
	R. Clark Robertson Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Spatiotemporal clustering is the process of grouping objects based on both their spatial and temporal similarity. This approach is useful when considering the distance between objects and how that distance changes through time. Spatiotemporal clustering analysis is applied to the maritime domain in this thesis, specifically to a defined area of water, during a period of time, in order to gain behavioral knowledge of vessel interactions and provide the opportunity to screen such interactions for further investigation. The proposed spatiotemporal clustering algorithm spatially clusters vessels in the water space using k-means clustering analysis, kinematically refines the clusters based on the similarity of vessel headings, speeds and the distance between them, and temporally analyzes the continuity of membership of the kinematic clusters through time to determine which clusters are moving. The algorithm is implemented in the MATLAB programming environment, verified with a synthetic data scenario, and validated with two real-world datasets.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AIS	Automatic identification system
C4	Command, control, communications, and cyber
CNO	Chief of Naval Operations
CONOPS	Concept of operations
DoD	Department of Defense
GNSS	Global navigation satellite system
GPS	Global positioning system
GUI	Graphical user interface
IMO	International Maritime Organization
KDVC	K-means, distance, and velocity clustering
MDA	Maritime domain awareness
MMSI	Maritime mobile service identity
MOC	Maritime operation center
M/V	Motor vessel
RMP	Recognized maritime picture
STC	Spatiotemporal clustering
UTC	Coordinated Universal Time
VOI	Vessel of interest

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Maritime domain awareness (MDA) presents a continuous challenge to national strategic decision makers and technical analysts. MDA is based on the concept that maritime security is achieved or improved through developing an understanding of events occurring in the maritime domain. The ability to autonomously classify vessel movement at sea in order to gain behavioral knowledge of a water space presents a significant challenge, especially when more than 90% of the world's commerce is conducted by sea, and non-traditional maritime threats, counter-proliferation, and piracy are increasingly more important to national security. Previously, much effort has been placed into the development of algorithms that employ time-series analysis to estimate vessel position and predict future vessel location and that determine what normal, and by extension, abnormal behavior of vessels at sea can be considered to be. These efforts support the creation and maintenance of the recognized maritime picture (RMP) but lack analysis of how ships interact and, by extension, a behavioral knowledge of a water space.

The objective of this thesis is to develop a method to autonomously analyze and classify ship movement and possible intent at sea in order to gain behavioral knowledge of a specific water space. The approach utilizes spatiotemporal clustering (STC) in which spatial relationships between objects are studied as they change over time. The STC concept has been applied to other areas, including urban combat environments, georeferenced mobile device tracking, traffic incidents, and the spread of the avian influenza H5N1. A spatiotemporal extension of STC algorithms designed to analyze urban combat environments is used as a method of classifying paralleling and following movement behavior in the maritime domain because vessels involved in illicit activity may exhibit these behaviors. The proposed STC algorithm for MDA application is modeled in the MATLAB programming environment, and results for both simulated and real-world scenarios are presented.

Although there are several methods to perform spatiotemporal analysis, in this thesis the process of first applying a k-means proximity filter, kinematically clustering

vessels at each unique time-step based on likeness of course, speed, and distance, and then performing temporal analysis, is proven to be effective in the maritime domain.

Both a simulation and real-world data analysis are presented using the MATLAB programming environment. The simulation scenario is presented to test operability of the kinematic clustering parameter thresholds of the STC algorithm and to verify functionality of text and visual outputs. Two real-world datasets, one taken from worldwide automatic identification system (AIS) position reports and the other from a global positioning system (GPS) source, are used with a hypothetical scenario to illustrate the possibilities revealed by this method of analysis.

In the first real-world data analysis, a hypothetical scenario is presented in which a vessel of interest (VOI) might have transferred illicit cargo on a specific date in the Strait of Malacca. On the date of interest there are approximately 5,000 AIS vessel position reports in the area of interest. The STC algorithm developed in this thesis is applied to the AIS dataset to characterize VOI interactions with other vessels in the area of interest and to determine if the VOI interacts with or moves in coordination with any other vessels. The results of this analysis provide a realistic understanding of the capabilities of the STC algorithm. The VOI is found to be spatiotemporally clustered with another vessel at multiple time steps during a nearly four hour timespan. The results of the real-world data analysis offer enough insight as to the vessel's interactions during the time period of interest for an analyst's further investigation.

The second real-world data analysis is performed on GPS data gathered over a twenty-three minute period via GPS transmitters mounted on six vehicles simulating a collection of small boats travelling together. The drive-plan for the experiment required three of the vehicles to form a convoy and maintain speed and distance for the duration of the exercise. The other three vehicles acted as confuser vehicles and moved in and out of the convoy. The analysis from the STC algorithm reveals that multiple moving clusters are tracked over the twenty-three minute period, which is in accordance with the drive-plan. The two real-world data analyses highlight two different possible uses for the STC algorithm.

ACKNOWLEDGMENTS

I would like to give my love and thanks to my wife, Katy, for her support, patience, commitment, and never-ending encouragement during the course of my master’s work, specifically for the days and nights on which I stared at the computer. I could not have been as successful in my work without you.

I would also like to extend my gratitude to Professor Jim Scrofani, Professor Murali Tummala, and Professor Dave Garren of the Naval Postgraduate School, Monterey, California, for their patience, instruction, guidance, and contributions to this work.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Maritime domain awareness (MDA) presents a continuous challenge to national strategic decision makers and technical analysts. Defined as “the effective understanding of anything associated with the maritime domain that could impact the security, safety, economy, or environment of a nation,” MDA is based on the concept that maritime security is achieved or improved through developing an understanding of events occurring in the maritime domain [1], [2]. A previous commander of the U.S. Sixth Fleet indicated that the interdiction of illegal or terrorist activity in today’s complex environment is utterly dependent on maritime domain awareness [3].

The gathering and sharing of information and intelligence, between international maritime and other partner organizations, in order to develop a recognized maritime picture (RMP), is a fundamental component of MDA [3]. In the past, the Department of Defense’s (DoD) MDA focus was on understanding opposing navies and maritime forces, but more recently this focus has shifted to commercial vessels, as non-traditional maritime threats, counter-proliferation, and piracy have become increasingly more important to national security [3]. Accordingly, these trends necessitate the tracking, managing, and understanding of extensive details about many more vessels, from merchant tankers to fishing trawlers and pleasure craft [3].

In addition to these emerging trends, the fact that more than ninety percent of the world’s commerce today is conducted via the sea results in an even greater need for capabilities to track vessels and assess their behaviors [4]. The ability to autonomously classify vessel movement at sea, in order to gain this behavioral knowledge of a specific water space, is a novel challenge. From a national security perspective, vessels of interest (VOI) are often found to be moving together in formation, following or paralleling, or converging for transfer of illicit cargo. From a simple navigation radar display, an example of which is illustrated in Figure 1, to a vessel listing system like the automatic identification system (AIS), classifying these types of movement utilizing the tools available is a time-intensive process for even a team of analysts. Automating the analysis of ship movements through the use of a spatiotemporal clustering algorithm provides a flexible, user-interfaced solution.

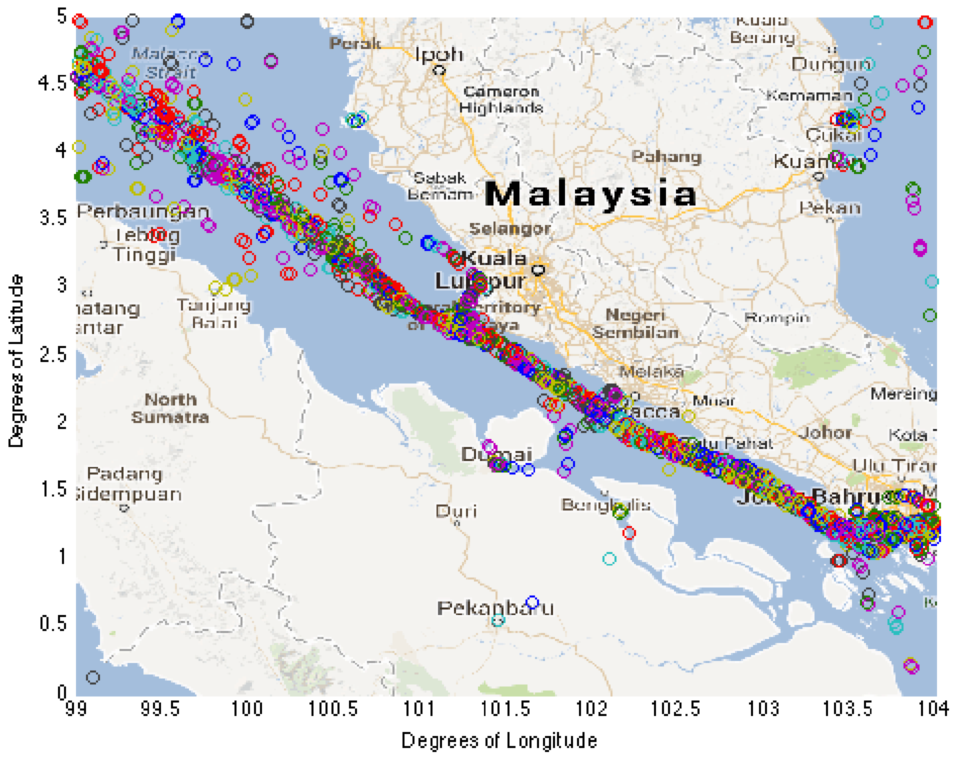


Figure 1. A 24- hour satellite AIS position report collection in the Strait of Malacca.

A. THESIS OBJECTIVE

The objective of this thesis is to develop a method to autonomously analyze and classify ship movement and possible intent at sea in order to gain behavioral knowledge of a specific water space. The behaviors focused on are paralleling and following because they are common interactions between vessels at sea and their traits are easily defined. Vessels involved in illicit activity may exhibit either of these behavioral traits.

A spatiotemporal clustering (STC) extension of the urban combat environment direction and displacement directivity algorithms presented in [5]–[7] is proposed in this thesis as a method of classifying paralleling and following movement behavior in the maritime domain. The proposed STC algorithm for MDA application is modeled in MATLAB, and results for both a synthetic simulation scenario and real-world dataset analyses are presented.

B. RELATED WORK

In the maritime domain much effort has been placed into the development of algorithms that employ time-series analysis to estimate vessel position, predict future vessel location, and determine what is normal, and by extension, abnormal behavior of vessels at sea [8]. Specifically, Tunaley describes the development of a ship detection and tracking program that is designed to analyze processed imagery rapidly and inexpensively and to deliver messages automatically by email [9]. Efforts like Tunaley's support the creation and maintenance of the RMP but lack analysis of how ships interact and, by extension, a behavioral knowledge of the water space. One potential solution to overcome this shortcoming is STC analysis as proposed in this thesis.

The STC concept has not previously been applied to the maritime domain but has been applied in other areas. Hwang et al. examined the properties of moving objects in road networks in [10] and defined spatiotemporal similarity between trajectories based on points and times of interest. From their work, early methods were proposed for searching for similar trajectories amongst moving objects in road networks, and one of these early methods serve as the framework for the STC analysis in this thesis. Si et al. later used STC via the space-time permutation scan statistic to analyze the spread of the avian influenza H5N1 in poultry, wild birds, and humans in [11]. Their STC analysis aided them in determining that H5N1 outbreaks showed a clear seasonal pattern and were shown to be relatable to the patterns of migration of wild birds. The space-time permutation model was an option for the basis of STC in this thesis, but the STC algorithm used in this thesis analyzes the water space of interest in one sweep instead of using multiple smaller area sweeps.

Yuan and Raubal expanded on the STC concept and provided a framework for the extraction of spatiotemporal knowledge from georeferenced mobile phone devices and other information and communication technologies [12]. In [13], Eckley and Curtin presented methods for performing spatiotemporal analysis, with special attention given to the interpretation of the results for traffic incidents and also presented arguments for performing spatial and temporal analyses independently. The works presented in [12] and

[13] expanded upon the use of STC to analyze data and explored the idea of extracting information from the analysis results, similar to the intent of the work in this thesis.

Das et al. created STC algorithms to explore urban combat environments in [5]-[7] to introduce a method of analyzing troop movements and interactions on land. Their use of STC in combat environments was the genesis of the idea to extend spatiotemporal analysis to the maritime domain to study the interactions of vessels at sea. An extension of the direction and displacement directivity algorithm presented in [5]-[7] is used in this thesis.

C. THESIS OUTLINE

The outline of this thesis is as follows: background information on several topics including the United States Navy MDA concept, an explanation of one of the data sources used in this thesis, spatiotemporal clustering and its fundamental parts, and k-means clustering are presented in Chapter II. The STC algorithm components of data conditioning, cluster preprocessing, proximity filtering, kinematic filtering, temporal analysis, and the post-processing of spatiotemporal results to create easily understood text and visual outputs are discussed in Chapter III. The set-up and results of the simulation scenario and real-world data analyses are detailed in Chapter IV. A summary of the work completed, the significant results, and ideas for future work are provided in Chapter V. The MATLAB code that implements the STC algorithm is included in the Appendix.

II. BACKGROUND

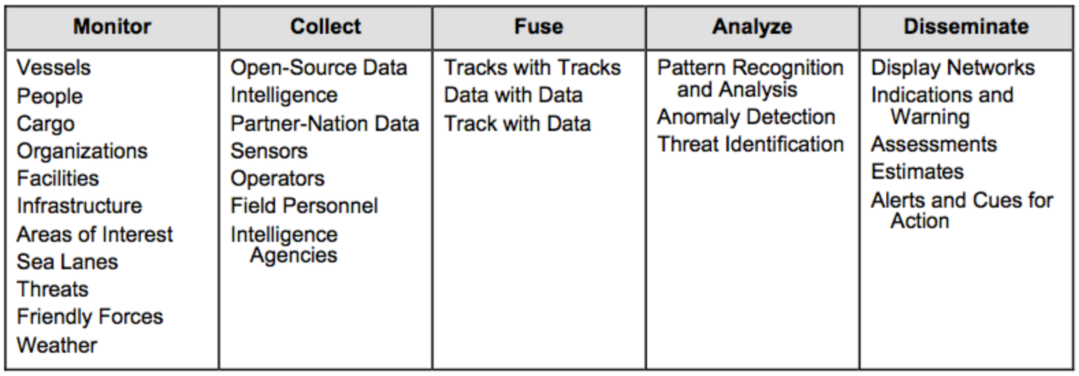
A brief overview of the DoD MDA concept was provided in Chapter I. In this chapter, a more detailed description of the Navy's approach to MDA is presented in order to frame the current methods being utilized and bring to light the shortfalls faced by these methods. A discussion of a typical data source used in real-world analysis is presented, along with alternative sources. The STC concept and spatial and temporal data are defined. The advantages and disadvantages of three methods of STC are discussed, and an overview of the k-means clustering algorithm is provided. At the end of this chapter, the reader will have the necessary understanding to comprehend the details of the STC algorithm prior to its application to synthetic and real-world datasets.

A. NAVY MARITIME DOMAIN AWARENESS CONCEPT

The Navy MDA concept was elaborated on in May 2007 by then Chief of Naval Operations (CNO) Admiral Michael Mullen. The memorandum he signed provided a framework to prioritize the MDA efforts across the Navy, to ensure alignment with external MDA initiatives, and to outline the fleet MDA concept of operations (CONOPS) [14]. The implementation of MDA guidance was intended to take ten years, so efforts to improve MDA are ongoing and remain relevant [14]. Vice Admiral Nancy Brown, then Director, Command, Control, Communications and Computers (C4) Systems on the Joint Staff, summarized the importance of the Navy's role in MDA in 2010 when she indicated that with the astronomical number of vessels at sea, the purpose of MDA is to gain an understanding of the small number of them that are involved in illicit activity, and that mission on the high seas is owned by the Navy [14].

The CONOPS for MDA organizes the MDA process into five activities. The activities are monitor, collect, fuse, analyze, and disseminate and are detailed in Table 1.

Table 1. The U.S. Navy fleet maritime domain awareness process. From [3].



Monitor	Collect	Fuse	Analyze	Disseminate
Vessels People Cargo Organizations Facilities Infrastructure Areas of Interest Sea Lanes Threats Friendly Forces Weather	Open-Source Data Intelligence Partner-Nation Data Sensors Operators Field Personnel Intelligence Agencies	Tracks with Tracks Data with Data Track with Data	Pattern Recognition and Analysis Anomaly Detection Threat Identification	Display Networks Indications and Warning Assessments Estimates Alerts and Cues for Action

MDA analysts and maritime operation centers (MOC) should have the ability to persistently monitor, access, and maintain information on vessels and craft, cargo, vessel crews and passengers, maritime infrastructure, and other identified areas of interest. Legally protected personal information and private sector proprietary information should be safeguarded and protected in accordance with U.S. and DoD regulations. The MOC should also be efficient in collecting, fusing, analyzing, and disseminating information to facilitate effective understanding and threat detection to decision makers and partner MDA organizations [3]. The introduction and development of a STC algorithm to support MDA seeks to support the Navy's CONOPS for the MDA mission as described above.

B. DATA SOURCE OPTIONS FOR STC ANALYSIS

The STC algorithm is designed to ingest data in a particular manner, taking into consideration the position, course, and speed of vessels at sea. The data sources for the real-world analyses presented in Chapter IV are derived from AIS and the global navigation satellite system (GNSS), the broad term for the global positioning system (GPS). AIS is used because of its availability and the ease with which it can be formatted and used. In order to employ the algorithm, three requirements for a data source must be met. The first is that the data must provide knowledge of georeferenced vessel position, ideally in degrees of latitude and longitude. A source would be compatible as long as degrees of latitude and longitude can be extrapolated. The second requirement is that the data must provide knowledge of the course and speed associated with each vessel. The third requirement is that each vessel position report include a time reference. With these three requirements met, other parameters such as the vessel identifier can be defined locally, as the global identifier need only identify individual vessels to the program user. Various other data sources could be used with this algorithm, including data derived from intelligence, surveillance, and reconnaissance systems, as long as the three requirements above are met.

A disadvantage of using AIS as the primary data source is that a particular VOI may not properly report position or even transmit its AIS signal. Further, because only certain classes of vessels are required to operate AIS, a VOI may not be outfitted with the system. In this thesis the assumption is made that a VOI will properly report its position because vessels that are involved in illicit activity will often attempt to act as normal as possible in order to detract attention from them. Another assumption is that many vessels that are of interest for national security purposes fit the criteria to need the use of AIS onboard. The use of GNSS as a data source in the second scenario provides an alternative to AIS. GNSS does not have the above disadvantages, nor do the assumptions above need to be made. Those criteria are outlined in Chapter IV.

C. SPATIOTEMPORAL CLUSTERING

Kisilevich et al. [15] define STC as the process of grouping objects based on their spatial and temporal similarity. STC is a relatively new approach to data mining and has been primarily used in mobile device and other location-based data tracking problems. In its application to the maritime domain, STC analysis is used to detect paralleling and following behaviors between vessels. Paralleling behavior is when one vessel traces the same or similar spatial pattern at the same time as another vessel but is offset in space [7]. Following occurs when one vessel traces the same or similar spatial route as another vessel at a later time [7]. To provide a better understanding of STC it is necessary to define what spatial and temporal datasets are and how they can be analyzed to create a STC result.

1. The Relationship Between Spatial and Kinematic Data

Spatial data contains information that describes the location of objects in relation to Earth, to each other, or to another frame of reference. A metric of distance is used when describing spatial relationships between objects. Several distance metrics exist, but two of the most common are the Manhattan and Euclidean distances. The Manhattan distance d_{MH} between two points (x_1, y_1) and (x_2, y_2) is computed as the sum of the absolute differences of their Cartesian coordinates and is mathematically described as [16]

$$d_{MH} = |x_1 - x_2| + |y_1 - y_2|. \tag{1}$$

This can be more easily thought of as the number of city blocks we would travel between points and is most relevant when measuring distance in a grid-constrained context [16].

Euclidean distance, on the other hand, is the more common straight-line distance metric and is defined as the length of the line segment that connects two points [16]. The Euclidean distance d_{EU} between two points (x_1, y_1) and (x_2, y_2) is mathematically described as [16]

$$d_{EU} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}. \tag{2}$$

To understand the underlying behaviors of paralleling and following, kinematic information is also required. Kinematics describes the motion of points in a particular frame of reference, and kinematic similarity between nearby objects can suggest collusion and other behaviors of interest [17]. In this maritime application, vessel characteristics like course and speed will factor largely into determining vessels that are exhibiting paralleling or following behavior. For this reason the STC algorithm does not exclusively spatially cluster vessels at sea, but rather does so kinematically by considering course and speed characteristics in addition to spatial relationships.

2. Temporal Data

Temporal data represents variable-state variation in time. It is often collected to analyze weather patterns, monitor traffic conditions, or study demographic trends, among many other applications. Any variable that changes over time can be organized into a temporal dataset. The fifth stage of the STC algorithm is temporal data analysis.

3. Spatiotemporal Clustering Approaches

As stated, in this thesis STC analysis is applied to the maritime domain in order to autonomously classify vessel movement at sea in order to gain behavioral knowledge of a water space, specifically to determine when vessels exhibit paralleling or following behavior as an indicator of possible illicit behavior.

Hwang, Kang, and Li [10] propose three methods to perform STC of objects moving on road networks. These methods can be extended to moving vessels at sea. In the first method, clusters are formed based on spatial clustering and are then refined with temporal analysis. In the second method, clusters are formed based on temporal clustering and then refined with spatial analysis. In the third method, clusters are formed based on the simultaneous consideration of spatial and temporal analyses.

In this thesis the first method is used as the general approach to STC. The second method is not easily translated to the maritime domain because temporal clustering would yield much larger clusters in need of kinematic refinement and, in turn, create a more difficult problem. This would not be an efficient approach to analyzing a water space that has high traffic density, like a major strait. The third method is likely the most robust of the three, but because of the time-latent nature of maritime domain data, it is not necessary to simultaneously consider space and time. The simultaneous accounting of kinematic and temporal attributes creates a larger workload and leads to inefficiencies in the STC algorithms' performance. Based on the first method above, vessels at sea are kinematically clustered and then refined through temporal analysis to form spatiotemporal results which can be processed for further understanding.

D. K-MEANS CLUSTERING ANALYSIS

A version of the k-means clustering algorithm was first proposed in 1957 as a technique for pulse code modulation, but the term "k-means clustering" came into use when McQueen [18] published the term in 1967 [19]. K-means clustering seeks to partition a set of n observations $\left(x_1, x_2 \ldots x_n\right)$ into a set c of k clusters $c = \left\{c_1, c_2 \ldots c_k\right\}$, where both the observations and clusters are vectors of specified dimension. The k-means clustering process can be described as the optimization of the objective function [20]

$$J = \underset{c}{\arg\min} \sum_{j=1}^{k} \sum_{x_i \in c_j} \left\| x_i - \mu_j \right\|^2 \tag{3}$$

where μ_j is the centroid of observations in c_j. The centroid of a cluster is defined as the mean of the observations in the cluster as

$$\mu_j = \frac{1}{N_{cj}} \sum_{i=1}^{N_{cj}} x_i, \tag{4}$$

where x_i are members of the cluster and N_{cj} is the number of observations in the cluster.

The most common form of the k-means algorithm uses an iterative refinement approach with two steps. In the assignment step an observation x_i is assigned to the cluster whose centroid is closest to it using the nearest neighbor rule

$$d\left(x_i, \mu_l\right) \le d\left(x_i, \mu_j\right) \text{ for } j \ne k \text{ and } 1 \le j \le k, \tag{5}$$

where the distance d is defined as

$$d\left(x_i, \mu_l\right) = \sqrt{\sum_{m} \left(\mu_{l_m} - x_{i_m}\right)^2}. \tag{6}$$

Once all observations have been assigned to a cluster, the update step calculates the new centroid of each cluster. The algorithm then goes back to the assignment step and reassigns the observations using the new cluster centroids. The algorithm is complete when the observation assignments no longer change [20].

The k-means clustering algorithm uses the Euclidean distance metric when minimizing the within-cluster sum of squares and calculating the mean in the assignment step. The main drawback to k-means clustering is that the number of clusters to be used k is an input parameter to the algorithm and a poor choice for k can lead to poor results. As such, any use of the k-means algorithm should include a discussion regarding the selection of k. An example of the result of k-means clustering is given in Figure 2, in which $k = 2$ and the data points associated with each cluster are colored red and blue.

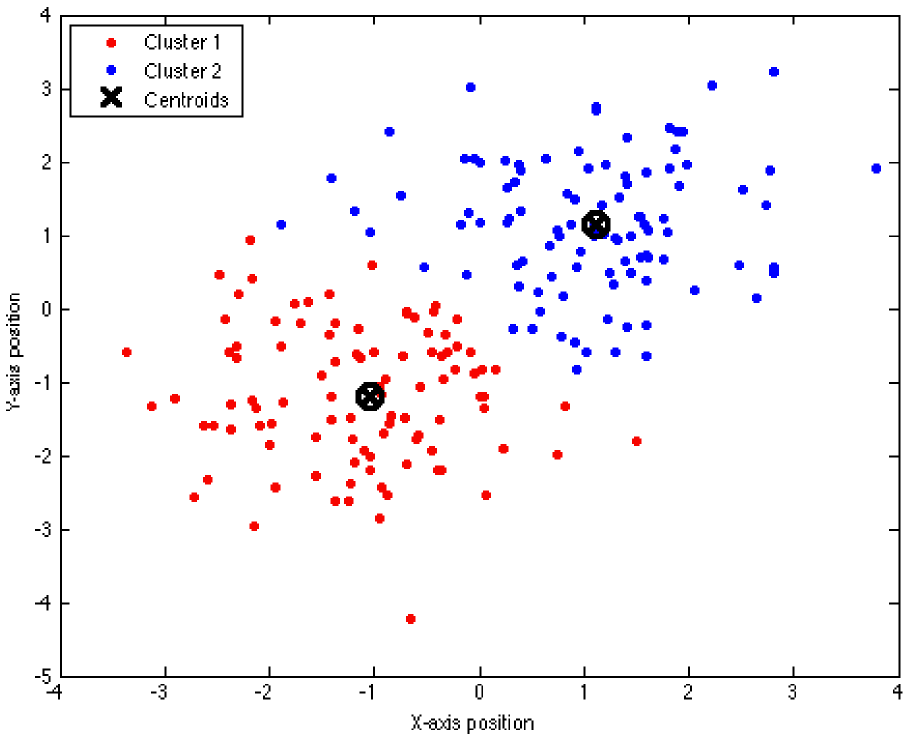


Figure 2. K-means clustering result on a random set of data points when *k*=2. After [21].

In this chapter an overview of the Navy's approach to MDA was presented to frame possible uses of the STC algorithm. A discussion of AIS data and alternative sources was discussed. Spatial and kinematic data were conceptualized, and an overview of the k-means clustering algorithm was provided.

THIS PAGE INTENTIONALLY LEFT BLANK

III. SPATIOTEMPORAL CLUSTERING ALGORITHM

A background of the STC concept was presented in Chapter II. Specifically, the relationship of spatial and kinematic data was addressed, and three methods of STC were presented. The method of forming clusters kinematically and performing temporal analysis to form spatiotemporal clusters is applied to the maritime domain in this chapter. An overview of the proposed algorithm is presented, followed by an in-depth explanation of the progression and development of the STC algorithm, with a focus on its six stages: 1) data conditioning, 2) cluster preprocessing, 3) proximity filtering, 4) kinematic filtering, 5) temporal refinement, and 6) post-processing of spatiotemporal results.

The algorithm begins with an input dataset consisting of vessel position reports during a specific timeframe. After user-provided input for parameter definitions, the data is submitted to a data conditioning stage in which time indices are assigned and the data is formatted, filtered, and converted. The proximity filtering stage spatially refines the position attributes of the vessel position reports, and through kinematic filtering, clusters of vessels are formed at each time-step based on similarity of vessel courses, speeds, and distance between them. The kinematic clusters are temporally refined to determine which of them can be considered moving clusters and which are only present at one instance in time. Once vessel position reports have been analyzed both kinematically and temporally, the results are spatiotemporal in nature and completely describe how the kinematic relationships between vessels change over time. Behavioral knowledge of the water space can then be gained through post-processing of the results and interpretation of the text and visual outputs. The progression of the six stages of the STC algorithm along with data input and behavioral knowledge output are depicted in Figure 3.

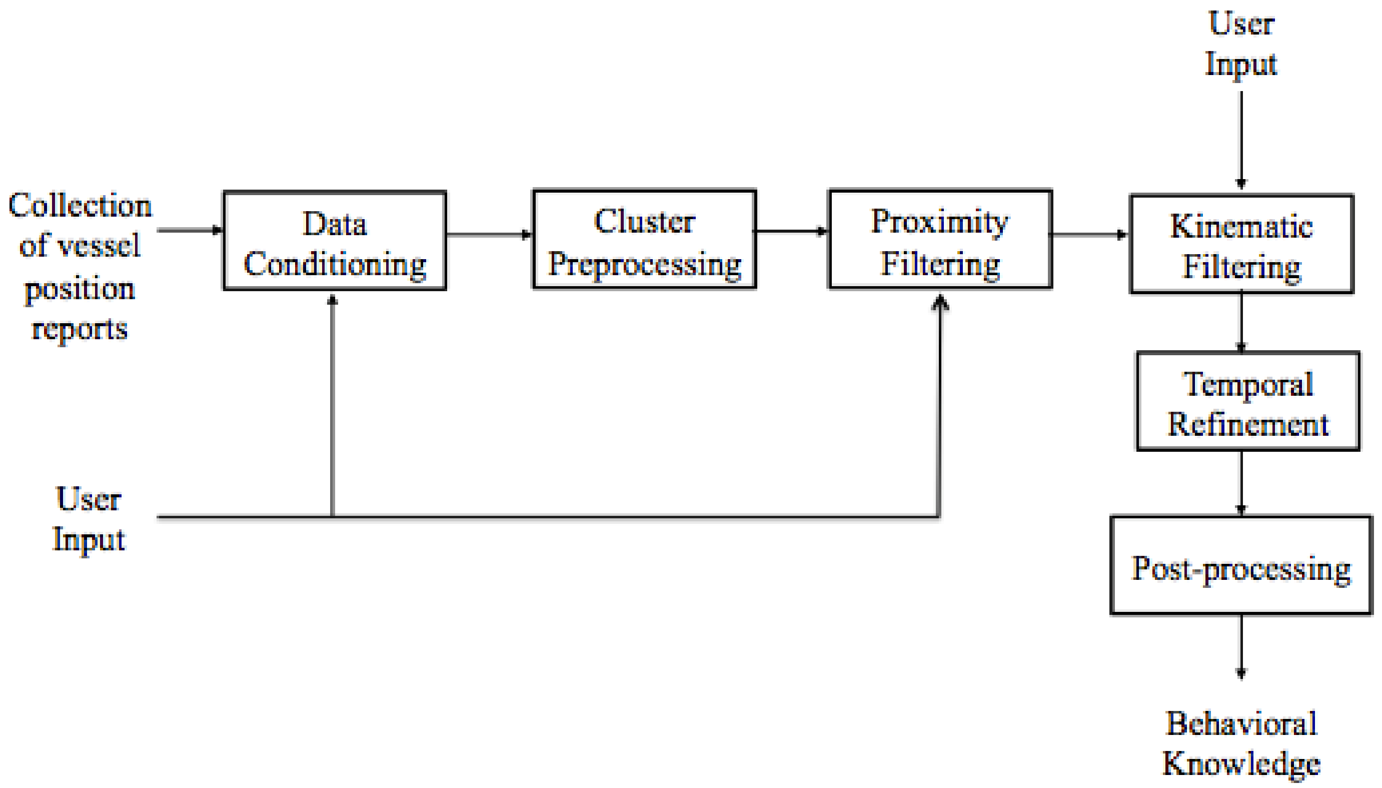


Figure 3. Top-level view of the spatiotemporal clustering algorithm.

A. INPUT

As illustrated in Figure 3, there are two types of input provided to the STC algorithm. The first is the collection of vessel position reports, which is the bulk data the STC algorithm analyzes to gain behavioral knowledge of the water space of interest. The second input is the user-provided input. Different parameters must be defined, either through use of default values or through manual input, to enable the various stages of the STC algorithm to function.

1. Collection of Vessel Position Reports

Vessel attribute data reports, which are transmitted by vessels as discussed in Chapter II, containing information on position, course, speed, a timestamp of the report, and other attributes this work does not concern, are collected from a suitable data source. A vessel identifier, such as MMSI, is typically supplied, and a local identifier may be defined, if necessary. A typical data source for this application is AIS information. In this thesis, AIS is the primary data source used because of its availability. The International Maritime Organization (IMO) requires AIS to be fitted aboard international voyaging ships with gross tonnage of greater than 300 tons and all passenger ships regardless of size [22]. The IMO requirement means that most sea-going commercial vessels are outfitted with and required to transmit position reports in AIS. AIS is, therefore, a suitable selection as the input data source for the development of a STC algorithm aimed at improving MDA. A typical AIS position report dataset is formatted into eleven fields, the titles of which are provided in Table 2.

The input dataset provided to the STC algorithm is a collection of AIS data for a specific period of time in a specific location. For the purposes of the STC algorithm, the data fields required are MMSI, speed over ground, longitude, latitude, course over ground, and the Coordinated Universal Time (UTC) timestamp. These six fields must be formatted and placed in the correct order to be analyzed and understood by the STC algorithm. This formatting occurs in two stages, data conditioning and cluster preprocessing, which are discussed following a description of user input.

Table 2. Typical field headings of an AIS vessel position and attribute report.

MID	MMSI
Nav Status	Rate of Turn
Speed Over Ground	Longitude
Latitude	Course Over Ground
Heading	UTC Timestamp
Source	

2. User Input

The algorithm requires four mandatory and five optional user inputs. The four mandatory user inputs are: 1) data source, 2) parameter selection method, 3) location filter definition, and 4) minimum vessel speed for consideration. The five optional user inputs, each of which has an associated program default value, are: 1) time window length, 2) maximum distance between contacts, 3) maximum heading difference between contacts, 4) maximum speed difference between contacts, and 5) moving cluster threshold value. Both the mandatory and optional user inputs are summarized in Table 3.

a. Mandatory User Inputs

(1) Data Source and Type

The first mandatory user-provided input is the data source positional data type. Positional data is provided in either degrees of latitude and longitude or in Cartesian coordinates. Positional data in degrees of longitude and latitude require conversion to the *x-y* coordinate grid for STC algorithm analysis, with zero degrees longitude and latitude serving as the origin. Conversely, data presented directly in the *x-y* coordinate grid are excused from the data conversion and preprocessing step. Both types of data then undergo time indexing in order to create time-step assignments.

Table 3. Summary of mandatory and optional user inputs with default values and selection ranges.

Variable	Mandatory/Optional	Default Value	Selection Range
Data Source and Type	Mandatory	-	Synthetic or Real-World
Parameter Selection Method	Mandatory	-	Default Values or Manual Input
Location Filter Definition	Mandatory	-	Central Latitude and Longitude, and Width and Height of Box
Minimum Vessel Speed	Mandatory	2 Knots	0 – 30 Knots
Time Window Length	Optional	12 Minutes	1 – 120 Minutes
Maximum Distance Between Contacts	Optional	8 Nautical Miles	0 – 50 Nautical Miles
Maximum Heading Difference	Optional	39°	0° to 359°
Maximum Speed Difference	Optional	16 Knots	0 – 30 Knots
Moving Cluster Threshold Value	Optional	60%	0% - 100%

(2) Parameter Selection Method

The second mandatory user-provided input required for program operability is the parameter selection method. When prompted, the user must decide whether to use program default values or to manually input them for STC algorithm calculations. The default values are explained in detail as part of each parameter's discussion that follows. Manual user inputs for each parameter have a high degree of freedom and are bound only by technical constraints (e.g., a vessel can only have a course heading between 0° and 359°).

(3) Location Filter Definition

The third mandatory user-provided input defines the location of interest for analysis. This input requires four attributes: 1) central latitude, 2) central longitude, 3) height, and 4) width. All four parts are defined in tenths of degrees of latitude and longitude and form a rectangular shaped bounding box. With this input the user can tailor the geographical region of interest for STC analysis.

(4) Minimum Vessel Speed

The fourth mandatory user-provided input is the minimum vessel speed for the STC algorithm to consider. The default value is two knots, but the user can select any speed value from zero to thirty knots. All contacts traveling at or below the selected speed are eliminated from consideration by the STC algorithm.

b. *Optional User Inputs*

The five optional user-provided inputs each have an associated default value. If the user opts for the manual parameter selection method, the following five parameters require specific assignment.

(1) Time Window Length

In order to make proper time-step assignments, the STC algorithm requires a defined time window length in minutes, as well as a defined start hour and start minute. The time window length serves as the minimum amount of time that can exist between each individual contact's position reports. If a contact reports itself more than once per time window, a function of the STC algorithm averages the multiple reports into one representative contact report. The values available for user selection range from one to one hundred and twenty minutes. The default setting is twelve minutes and is derived from the six minute rule of navigation, which says that the speed at which a vessel is traveling divided by ten is the distance it will travel in six minutes and is defined as

$$d = \frac{s}{10} \tag{7}$$

where d is the distance in nautical miles traveled in six minutes, and s is the speed in knots of the vessel [23]. The start hour and start minute are defined by the user as the starting time from which to calculate time steps based on time window length. For example, if the start hour is 01 and the start minute is 00, and time window length is twelve minutes, time-step one will be from 01:00:00-01:11:59 and time-step two from 01:12:00-01:23:59.

(2) Maximum Distance Between Contacts

During the kinematic clustering of contacts there are four factors taken into consideration. The first parameter input to kinematic clustering is the maximum distance between contacts measured in nautical miles. The value of this parameter is subjective as to how far apart two vessels can be and be considered to be paralleling or following each other in a manner of interest. Two logical values for this parameter are line-of-radar-sight and line-of-visual-sight, which are the distances from which a vessel can be detected by radar and visually, respectively, and can be calculated as

$$R_{radar} = 1.23\left(\sqrt{h_1} + \sqrt{h_2}\right) \tag{8}$$

$$R_{visual} = \sqrt{\frac{h_1}{0.5736}} \tag{9}$$

where R is the range in nautical miles, h_1 and h_2 are the height of the observer and target in feet, respectively [24].

When calculated with h_1 and h_2 set to the height of the pilothouse of a DDG-51 Arleigh Burke Class Destroyer, the values for R_{radar} and R_{visual} are approximately eight and 16 nautical miles, respectively. The default value for maximum distance between contacts is set to eight nautical miles. The reason for this is that while ships not in visual sight of each other could be exhibiting behavior of interest, illicit behavior is more likely to occur when ships are within visual sight.

(3) Maximum Heading Difference

The next parameter input is the maximum heading difference, measured in degrees between contacts. When determining the value of maximum heading difference to consider, the user subjectively defines what is encompassed in paralleling or following behavior. The range of input from which the user can select begins at 0° and extends to 359°. A key consideration when setting the value for this parameter is the distance between the two contacts at the start of the analysis. The default setting for this parameter is 39° and is based on the contacts beginning essentially in the same position. This value was determined as the angle that is formed for two vessels that are slightly more than eight nautical miles apart, the default value for maximum distance between contacts, after traveling for sixty minutes, or five time steps, assuming the default value for time window length, at a speed of 12 knots. The setup of this is illustrated in Figure 4. The values for maximum heading difference when using alternate starting distances between contacts us shown in Table 4.

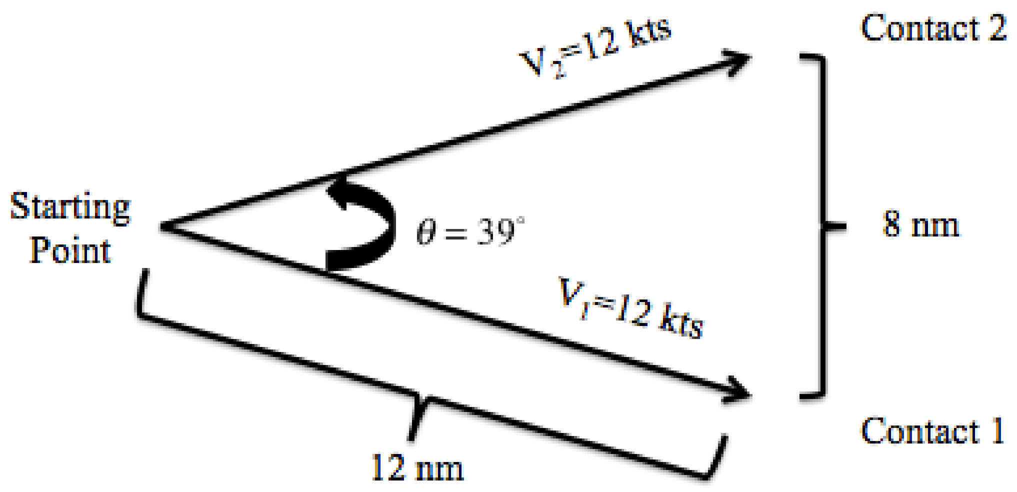


Figure 4. An illustration of the determination of the maximum heading difference of two contacts beginning in the same position.

Table 4. Maximum heading difference values for various beginning distances between two contacts traveling at 12 knots for sixty minutes.

Starting Distance Between Contacts	**Maximum Heading Difference**
0 Nautical Miles	39°
1 Nautical Mile	34°
3 Nautical Miles	24°
5 Nautical Miles	14°

(4) Maximum Speed Difference

Similar to the maximum heading difference, the maximum speed difference between contacts is a key parameter during the kinematic clustering stage of the algorithm. The range of input for maximum speed difference is from zero to thirty knots. Selecting a value of zero or near zero indicates that for vessels to be clustered together they would need to be traveling at identical speeds. On the other hand, selecting a value near thirty, which is considered the higher end of maximum attainable speed of ships at sea, would result in this parameter not having a significant role in the clustering algorithm. The default for maximum speed difference is set to 16 knots, which was determined by considering the following overtaking problem, illustrated in Figure 5. Vessel A begins eight nautical miles astern of vessel B and is traveling 16 knots faster. After sixty minutes, vessel A is 8 nautical miles ahead of vessel B.

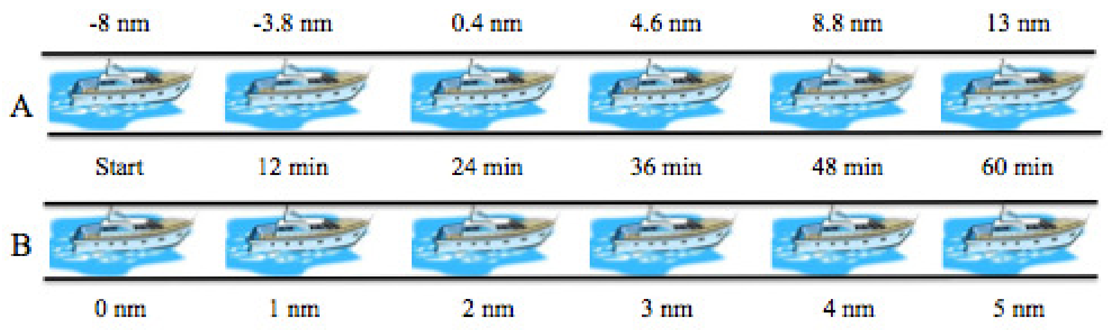


Figure 5. An illustration of the overtaking problem to determine the maximum speed difference between contacts that begin eight nautical miles apart.

(5) Moving Cluster Threshold Value

The moving cluster threshold value that is used to determine whether kinematic clusters are stored as moving clusters, or are discarded, must be defined. In this selection, the user defines the degree to which the membership of a moving cluster can change and still be considered a moving cluster. The sizes of the dataset and moving cluster have a large impact on the moving cluster threshold value selection. The larger the sizes, the higher the moving cluster threshold value should be. If the sizes are small, the moving cluster threshold value should be set to a smaller value to allow for better tracking. For example, if the moving cluster contains four contacts, the threshold could be set at 50% or below to track the cluster even if two vessels depart. If the cluster contains ten contacts, the threshold could be set to 80% to still allow for two vessels to depart the cluster. The range of values for selection begins at 0% and extends to 100%. A selection of 0% would discount the temporal consideration of the STC algorithm, while a selection of 100% would require that a moving cluster not lose any members in order to maintain its status as a moving cluster. The default value is 60%, which allows a moving cluster of five contacts to lose up to two members and still be considered a moving cluster.

B. DATA CONDITIONING

Data conditioning is the stage in which the input dataset is formatted and aligned for STC algorithm use. First, time indexing, the process of converting UTC seconds to a time-step assignment, an integer greater than or equal to one, is applied to the input dataset. UTC seconds are converted to the time and date of the timestamp, and then, with the user input for start hour, start minute, and time window length, the time and date is converted to a time-step value t. A function designed to ensure that each MMSI or vessel identifier has only one position report for each time-step is also applied. If a vessel has more than one report in a time-step, the vessel attributes from the multiple reports are averaged to form one representative report for the time-step.

Second, the dataset is formatted and placed in the correct order so that the required data fields are properly aligned for STC algorithm use. Each unique AIS vessel attribute report is stored as a row in the matrix Ψ in which each column represents a vector of different attribute from the AIS data as

$$\Psi = \begin{bmatrix} x & y & r & s & m & t \end{bmatrix}, \tag{10}$$

where Ψ is of size $L_{AIS} \times 6$, and L_{AIS} is the number of AIS vessel reports in the input dataset. The variables are all vectors of length $L_{AIS} \times 1$ and x and y describe each vessel's position on the *x-y* coordinate grid, r its course, s its speed, m its MMSI, and t its time-step assignment.

C. CLUSTER PREPROCESSING

Cluster preprocessing is the introduction of vessel identity indexing and the decomposition of Ψ into manageable pieces for STC algorithm analysis. During the clustering preprocessing stage, each unique MMSI in the vector m is assigned an index q beginning at 1 and continuing until each unique MMSI has an assignment. The index q will be used as a subscript to identify which vessel's attributes are being used.

The Ψ matrix is decomposed by time-step assignment into a three-dimensional matrix ψ of dimension $L_t \times 4 \times N_t$ where N_t represents the maximum number of time steps, L_t is the number of vessels in each time-step, and $\sum L_t = L_{AIS}$. The data matrix ψ can be represented at the t-th time-step as

$$\psi^t = \begin{bmatrix} x' & y' & r' & s' \end{bmatrix}, \tag{11}$$

where

$$x' = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_q \end{bmatrix}, \; y' = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_q \end{bmatrix}, \; r' = \begin{bmatrix} r_1 \\ r_2 \\ . \\ . \\ r_q \end{bmatrix}, \text{ and } s' = \begin{bmatrix} s_1 \\ s_2 \\ . \\ . \\ s_q \end{bmatrix}, \tag{12}$$

and q is the vessel identity index.

D. PROXIMITY FILTERING

Proximity filtering, accomplished through the use of k-means clustering, identifies spatial relationships between vessels at a particular time-step. Only x and y position data of ψ^t is considered. The optimum number of centroids k available for assignment is calculated from the length of Ψ^t at the time-step being analyzed as [25]

$$L_t = 2k\log(k). \tag{13}$$

To determine the specific centroids at each time-step, the farthest-first methodology is used [25]. The x and y position information for the contact in the first row of Ψ^t is designated as the initial centroid. The second through k-th centroids are chosen to be the contact that has the maximum Euclidean distance from the previously selected centroid. For example, for $t = 1$ the first contact of ψ^1 is designated as the first centroid. The second centroid is chosen as the contact has the greatest distance from the first centroid, where distance is calculated as

$$d = \sqrt{(x_n - x_1)^2 + (y_n - y_1)^2} \quad \text{for } n = 2, 3, \ldots L_t. \tag{14}$$

The third centroid is chosen as the point that has the greatest distance d from the second centroid and so on.

Then the centroids are used to perform k-means clustering on the position data in ψ^t to spatially refine the contacts by assigning each contact to one of the centroids. The contacts associated with each centroid j at each time-step t are stored in a matrix c_j^t of size $L_{t_j} \times 4$ where L_{t_j} is the number of contacts assigned to each centroid at each time-step as

$$c_j^t = \begin{bmatrix} x' & y' & r' & s' \end{bmatrix} \text{for } j = 1, 2, \ldots k, \tag{15}$$

where x', y', r', and s' are vectors of size $L_{t_j} \times 1$ containing the respective attributes for the vessels in the cluster. After all contacts have been assigned to a centroid, k-means cluster assignments for a time-step are stored in a cell array ξ^t as

$$\xi^t = \left\{ c_j^t \quad c_{j+1}^t \quad . \quad . \quad . \quad c_k^t \right\}. \tag{16}$$

A cell array is a set of matrices of different dimensions. At the completion of the proximity filtering stage there is a cell array ξ^t for each time-step that contains the k-means assignment for all contacts in that time-step.

E. KINEMATIC FILTERING

Kinematic filtering is used to further refine the clustering results from the proximity filter to form clusters of contacts with attributes that exhibit paralleling or following behavior as defined by the user. In kinematic filtering, the similarities of course and speed attributes of vessels, as well as the distance between them, are determined for each k-means cluster. In each cluster c_j^t in ξ^t, a seed contact σ is determined as the vessel with the maximum speed in c_j^t to initialize kinematic filtering as

$$\sigma = \max(s'), \tag{17}$$

where s' is a vector of speeds in c_j^t. To formulate kinematic clusters, three thresholds are defined. The heading difference between the seed contact and all other contacts in c_j^t is defined as

$$T_H = \left| r_{q_\sigma} - r_{q_p} \right| \quad \text{for p} = 2,3,\ldots L_{t_j}, \tag{18}$$

the speed difference is defined as

$$T_S = \left| s_{q_\sigma} - s_{q_p} \right|, \tag{19}$$

and the distance between them is defined as

$$T_D = \sqrt{\left(x_{q_\sigma} - x_{q_p}\right)^2 + \left(y_{q_\sigma} - y_{q_p}\right)^2}. \tag{20}$$

If T_H, T_S, and T_D are less than the user defined thresholds for maximum heading difference, maximum speed difference, and maximum distance between contacts, the contacts associated with q_σ and q_p are classified as a kinematic cluster ϕ_l^t and are represented as

$$\phi_l^t = \left\{ \begin{array}{c} q_\sigma \\ q_p \end{array} \right\} \text{for } l = 1,2,\ldots L_t \tag{21}$$

where i is the number of kinematic clusters for each time-step of ξ^t. The length of ϕ_l^t varies and is dependent on the number of contacts that when compared with the seed contact result in T_H, T_S, and T_D values that are less than the user defined thresholds. The contacts assigned to ϕ_l^t from c_j^t are removed from further consideration for assignment to another kinematic cluster.

A new seed contact is determined so that T_H, T_S, and T_D may be calculated for the remaining contacts in c_j^t to discover other kinematic clusters as described in Equations (17)-(21). The iterative process continues until all contacts in c_j^t have been considered for assignment to a kinematic cluster. Upon completion, a snapshot of the kinematic clusters in ξ^t are compiled in a cell array β^t for temporal analysis as

$$\beta^t = \left\{ \begin{array}{ccccc} \phi_l^t & \phi_{l+1}^t & . & . & . & \phi_{L_t}^t \end{array} \right\}. \tag{22}$$

An example of a k-means, distance, and velocity kinematic clustering result is illustrated in Figure 6 in which there are four kinematic clusters identified at time-step two of a generic simulation. The maximum distance between contacts was defined as four nautical miles for the simulation, and each of the four resultant clusters are displayed in a different color. The contacts of each cluster were initially grouped by proximity filtering and further refined kinematically by determining similarities between vessel headings, speeds, and the distance between them.

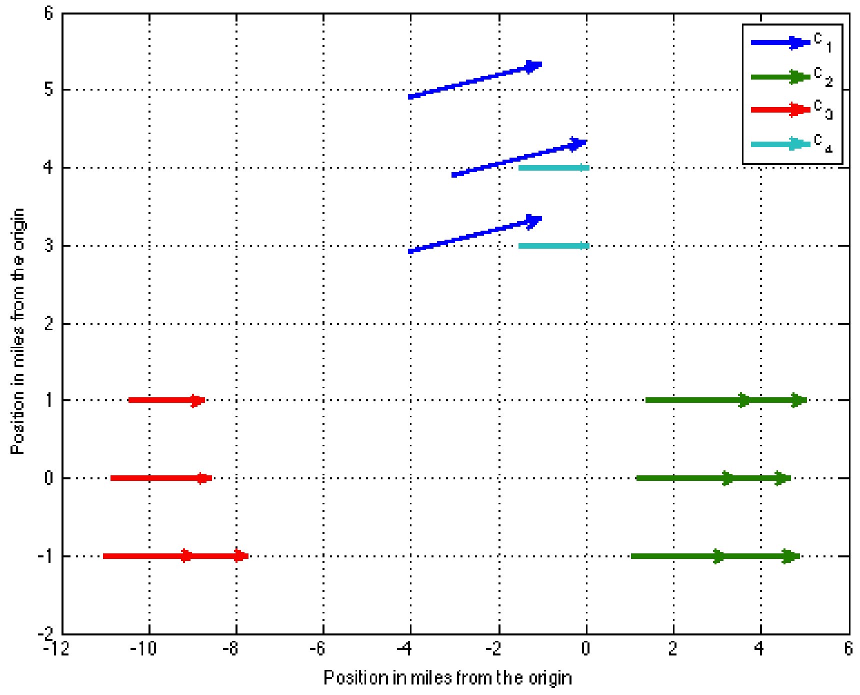


Figure 6. An example of k-means, distance, and velocity kinematic clustering results at time-step two of a generic simulation.

F. TEMPORAL REFINEMENT

Temporal refinement determines which kinematic clusters are moving in time and which are not. The input data to this stage are cell arrays that contain the membership of the kinematic clusters for each time-step. To determine which kinematic clusters are moving in time, it is necessary to calculate the similarity of the membership of the kinematic clusters in consecutive time steps. The first kinematic cluster in β^t, ϕ_l^t, is designated as the reference cluster, and the intersection of ϕ_l^t and each kinematic cluster in β^{t+1} is calculated as

$$I = length\left(\phi_l^t \cap \phi_v^{t+1}\right) \text{ for } v = 1,2,...L_{\beta^{t+1}}. \tag{23}$$

If an intersection is found between ϕ_l^t and a cluster in β^{t+1}, the variable fuse f is calculated as

$$f = \frac{I}{length\left(\phi_l^t\right)}, \tag{24}$$

and the next cluster in β^t, ϕ_{l+1}^t, becomes the reference cluster.

If an intersection is not found between ϕ_l^t and a cluster in β^{t+1}, the intersection is calculated between ϕ_l^t and each kinematic cluster in β^{t+2} as in Equation (23). If an intersection is found, fuse is calculated as in Equation (24). If an intersection is not found, ϕ_l^t is removed from consideration as a moving cluster and the second cluster in β^t, ϕ_{l+1}^t becomes the reference cluster, and the process is repeated.

Once all kinematic clusters in β^t have served as the reference cluster for intersection calculations, the kinematic clusters of β^{t+1} beginning with ϕ_l^{t+1} become the reference clusters. This iterative process continues until all the clusters in $\beta^{\max(t)-1}$ have served as the reference cluster. Kinematic clusters in $\beta^{\max(t)}$ are not considered in temporal refinement due to the lack of future kinematic clusters with which to calculate intersections.

When a value other than zero is calculated for f, it is compared to the user-defined moving cluster threshold value. If f is greater than or equal to that value, the associated kinematic cluster ϕ_l^t is stored to the cell array δ as

$$\delta = \left\{\phi_l^t\right\}, \tag{25}$$

where δ is of size $1 \times L_{MC}$ and L_{MC} is the number of moving clusters determined in temporal refinement.

G. POST-PROCESSING OF SPATIOTEMPORAL RESULTS

Upon completion of temporal analysis, three cell arrays, each containing spatiotemporal results, are constructed. Through post-processing of these cell arrays, it is possible to build usable outputs to determine information about the selected water space, including the detection of moving clusters of vessels that are exhibiting paralleling or following behavior and the ability to track vessel interactions over time.

1. Identifying Members of a Moving Cluster

The first cell array δ represents all kinematic clusters that move through time for a minimum of two time steps with a minimum threshold of continuous membership in the variable fuse f as

$$\delta = \begin{Bmatrix} \phi_l^t & \phi_{l+1}^t & \phi_{L_t}^t & . \\ \phi_{l+1}^t & \phi_{l+2}^t & \phi_{l+3}^t & \phi_{L_t}^t \\ . & . & . & . \\ \phi_l^z & \phi_{l+1}^z & \phi_{l+4}^z & . \end{Bmatrix} \tag{26}$$

where z is the maximum time-step that contains a moving cluster. The row location of any cluster ϕ in the cell array σ indicates the time steps at which the moving cluster exists. For example ϕ_{l+2}^t that appears in the second row, is a moving cluster that begins at time-step two.

Further analysis of this cell array provides details such as when a moving cluster forms, which contacts are members of the subject moving cluster, if contacts join or leave the cluster, and when, if ever, the cluster's membership becomes low enough that it can no longer be considered a moving cluster.

2. Kinematic Clusters Occurring at the Last Time-step of Data

The cell array ρ is equal to β^t at the last time-step of available data and is of size $1 \times L_t$ where L_t is the number of kinematic clusters found in β^t. Due to a lack of future time steps with which to compare this time-step of data, these current clusters are stored as a matter of interest to the user as

$$\rho = \left\{ \begin{array}{ccccc} \phi_l^{\max(t)} & \phi_{l+1}^{\max(t)} & . & . & \phi_{L_t}^{\max(t)} \end{array} \right\}. \tag{27}$$

The clusters in ρ are not known to be moving clusters but represent only the kinematic clusters of the last time-step of data. Information detailed in ρ is displayed only in the text output, under the "Moving Clusters" section, to avoid unnecessary cluttering of the visual output.

3. Generation of the Contacts of Interest List

The third cell array, θ, is the same size as δ, but rather than containing cluster membership information, it contains the global identity index q of contacts that either join or leave clusters. Vessels that move together as a moving cluster for several time steps are of interest to an analyst, but a vessel that joins a cluster then departs it, and then joins another cluster, is also of interest. The information contained in θ is displayed only in the text output, under the "Contacts of Interest" section.

4. Post-processing of Spatiotemporal Results to Form Usable Outputs

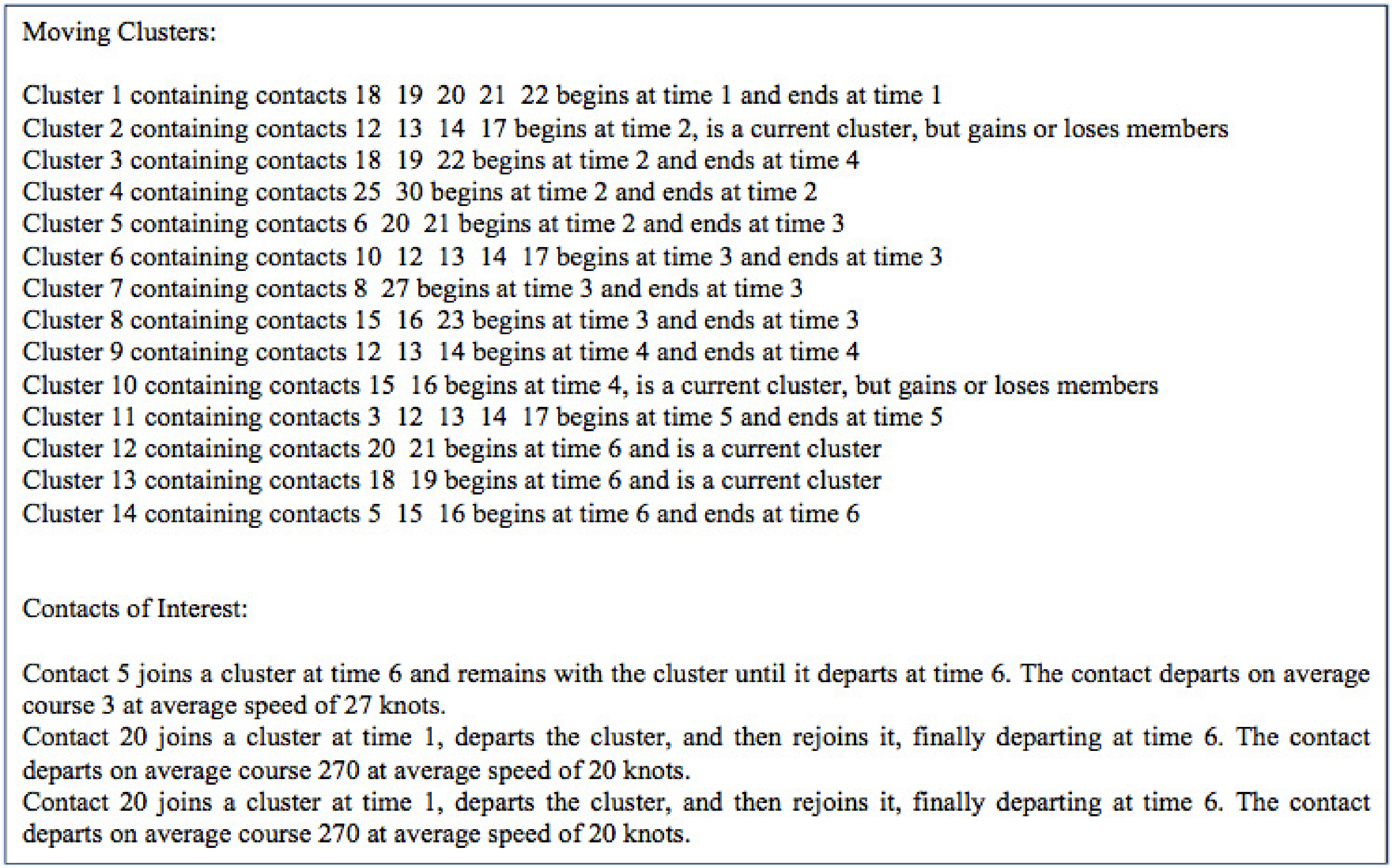
The post-processing of the spatiotemporal results stored in the cell arrays δ, ρ, and θ is the transfer of data into usable outputs for better user understanding. Post-processing of the cell arrays results in the generation of two outputs. One output is text based and the other is a visual representation of moving vessel clusters in the water space.

a. Text Output

The text output has two distinct parts, *Moving Clusters* and *Contacts of Interest*. The *Moving Clusters* part provides the global identifiers of the members of each moving cluster, as well as the time at which the cluster is formed and the time at which the cluster ceases to exist. Current clusters are also detailed in this manner in the text output. The second part of the text output, *Contacts of Interest*, results from the ρ cell array and contains the global identifier of any contact that joins or leaves an existent moving cluster, as well as the time at which the event occurs. If the contact is deemed to be leaving a moving cluster, the text output also details the heading and speed of the contact's departure. If there are two or more contacts that depart a larger moving cluster and continue to move as a smaller moving cluster, then the heading and speed reported is an average of the two contacts. An example of both parts of the text output is presented in Figure 7 where there are fourteen moving clusters identified and three contacts of interest defined.

b. Visual Output

The visual based output is an interactive, MATLAB-generated plot that depicts the position and track of each moving cluster analyzed by the STC algorithm. In Figure 8, there are several moving clusters displayed, some red, some orange, and some green, each color indicating the membership of the moving clusters as defined in Table 5. The position, heading, and speed of each moving cluster displayed is computed by averaging its constituent members' respective attributes. For example, for a given moving cluster, the speed reported is the average speed of all vessels contained within that specific cluster. Moving clusters are marked by an 'x' for each time-step that they occur and with an 'o' at their final time-step of occurrence. If a moving cluster is only present for two time steps, it is represented by an 'o'. The markers and track lines on the plot are color-coded to detail the degree of membership that the moving cluster maintains.



```
Moving Clusters:

Cluster 1 containing contacts 18  19  20  21  22 begins at time 1 and ends at time 1
Cluster 2 containing contacts 12  13  14  17 begins at time 2, is a current cluster, but gains or loses members
Cluster 3 containing contacts 18  19  22 begins at time 2 and ends at time 4
Cluster 4 containing contacts 25  30 begins at time 2 and ends at time 2
Cluster 5 containing contacts 6  20  21 begins at time 2 and ends at time 3
Cluster 6 containing contacts 10  12  13  14  17 begins at time 3 and ends at time 3
Cluster 7 containing contacts 8  27 begins at time 3 and ends at time 3
Cluster 8 containing contacts 15  16  23 begins at time 3 and ends at time 3
Cluster 9 containing contacts 12  13  14 begins at time 4 and ends at time 4
Cluster 10 containing contacts 15  16 begins at time 4, is a current cluster, but gains or loses members
Cluster 11 containing contacts 3  12  13  14  17 begins at time 5 and ends at time 5
Cluster 12 containing contacts 20  21 begins at time 6 and is a current cluster
Cluster 13 containing contacts 18  19 begins at time 6 and is a current cluster
Cluster 14 containing contacts 5  15  16 begins at time 6 and ends at time 6


Contacts of Interest:

Contact 5 joins a cluster at time 6 and remains with the cluster until it departs at time 6. The contact departs on average
course 3 at average speed of 27 knots.
Contact 20 joins a cluster at time 1, departs the cluster, and then rejoins it, finally departing at time 6. The contact
departs on average course 270 at average speed of 20 knots.
Contact 20 joins a cluster at time 1, departs the cluster, and then rejoins it, finally departing at time 6. The contact
departs on average course 270 at average speed of 20 knots.
```

Figure 7. An example text output from the STC algorithm's analysis of a generic simulation scenario.

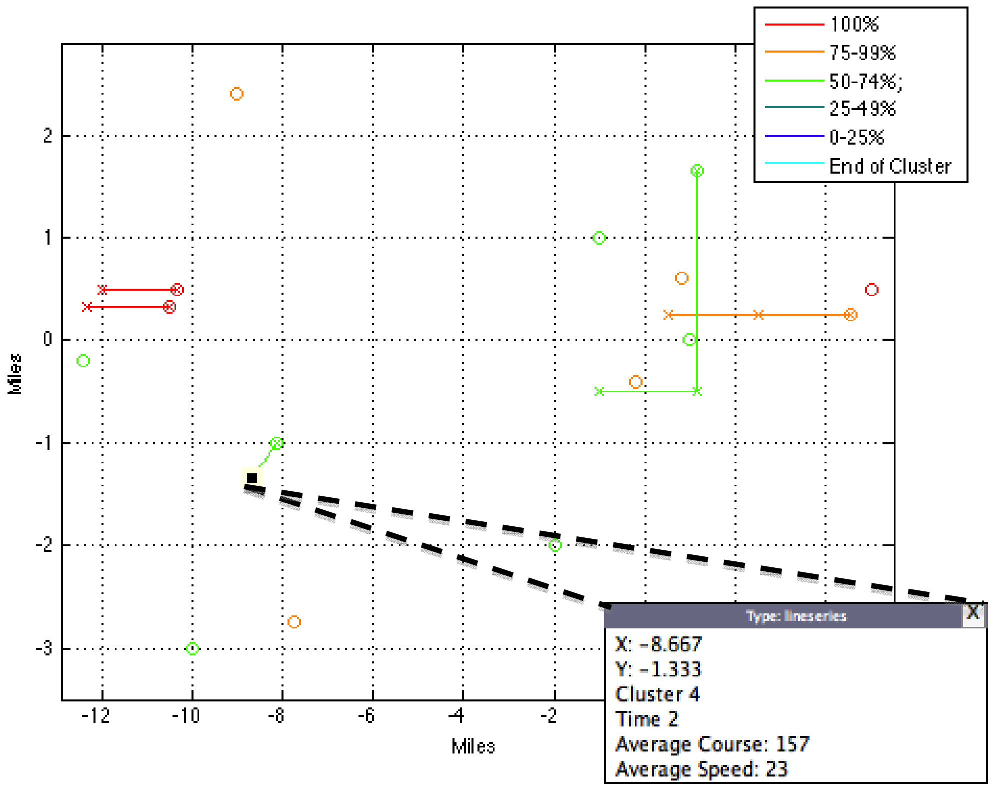


Figure 8. An example visual representation output of the STC algorithm's analysis of a generic simulation scenario.

Table 5. The MATLAB interactive plot color-code key for moving cluster membership identification.

Display Color	Percentage Membership Indicated
Red	100%
Orange	75% - 99%
Green	50% - 74%
Blue	25% - 49%
Violet	0% - 24%

The MATLAB plot also has an interactive functionality that is illustrated in Figure 8. The user is able to click on any marker of a moving cluster to determine which cluster is represented, at which time-step the cluster occurs, as well as the cluster's position, heading, and speed. In Figure 8 it is determined that moving cluster 4 exists at time-step two, in position $(-8.67, -1.33)$ heading 157° at 23 knots. To discover which contacts compose moving cluster 4, it is necessary to reference the first portion of the text output. Upon doing so, it can be determined that the contacts in moving cluster 4 have vessel identifiers of 25 and 30.

In this chapter, the six stages of the STC algorithm were detailed. Vessel data and user-provided input were discussed as the primary inputs to the STC algorithm. The data conditioning and cluster preprocessing stages, where datasets are time indexed, formatted, and aligned for algorithm use, were discussed. Proximity and kinematic filtering were defined and the attributes each considers were given. Temporal refinement to determine which clusters move through time was detailed, and examples of the usable outputs formed from post-processing of spatiotemporal results were presented. In Chapter IV, the STC algorithm is verified against a synthetic dataset and is validated using two real-world datasets.

IV. IMPLEMENTATION AND RESULTS

The STC algorithm presented in Chapter III is implemented, verified, and validated in this chapter. The algorithm is implemented in the MATLAB programming environment. To verify operability and validate its possible use, the STC algorithm is tested using synthetic and real-world datasets. An overview of the MATLAB functional code and user environment is provided, followed by a detailed explanation of the synthetic and real-world scenarios and results.

A. MATLAB IMPLEMENTATION MODEL

The MATLAB programming code is organized into a top-level function, Thesis, which calls various functions to execute the STC algorithm based on user input. An overview of the highest-level MATLAB functions and their purposes is found in Table 6. The detailed code for these and other functions is included in the appendix.

1. MATLAB Graphical User Interface

The MATLAB user interface for operation of the STC algorithm was designed to be intuitive and to ensure ease of use. The algorithm's user-provided inputs discussed in Chapter III are provided to the algorithm via the MATLAB graphical user interface (GUI) tool, examples of which are illustrated in Figures 9 and 10.

2. Simulation and Real-world Analysis Parameters

The thresholds used for the simulation and real-world analyses are defined in Table 7 and are the default values as discussed in Chapter III.

Table 6. MATLAB top-level functional organization of the STC algorithm.

Function	Overview
Thesis	Thesis is the function call to access spatiotemporal algorithms that support maritime domain awareness. The user will select whether their input is "Synthetic" or "Real-world," and the program will determine which set of functions to use for analysis.
RealWorld	RealWorld is the top-level function call for real-world AIS data analysis. This function begins with data pre-processing and filtering, and includes all functions required for spatiotemporal analysis of the water space.
KDVC	KDVC(A,clusters,time,climit,dlimit,spdlimit) returns the kinematic clusters at a given time-step based on user inputs where A is the input dataset, clusters is the matrix for output storage, time is the time-step being evaluated, climit is the user defined heading difference between contacts in degrees, dlimit is the user defined distance between contacts in nautical miles, and spdlimit is the user defined speed difference between contacts in knots.
Kinematic	Kinematic(A,c,climit,dlimit,spdlimit,h) returns the kinematic clusters at a given time-step based on user inputs where A is the input dataset, c is the seed contact, climit is the user defined heading difference between contacts in degrees, dlimit is the user defined distance between contacts in nautical miles, spdlimit is the user defined speed difference between contacts in knots, and h is the time-step.

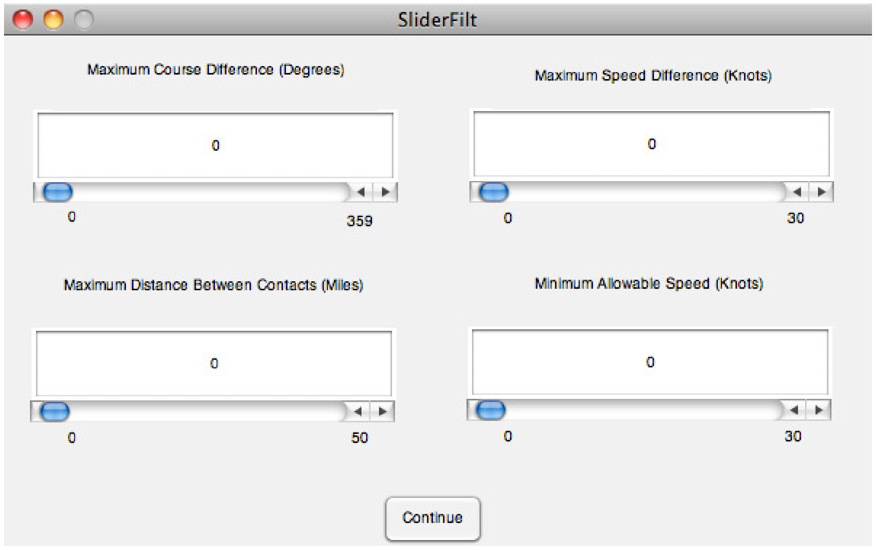

SliderFilt
Maximum Course Difference (Degrees)
0
0
359
Maximum Speed Difference (Knots)
0
0
30
Maximum Distance Between Contacts (Miles)
0
0
50
Minimum Allowable Speed (Knots)
0
0
30
Continue

Figure 9. The MATLAB graphical user interface with slider and open text parameter input options.

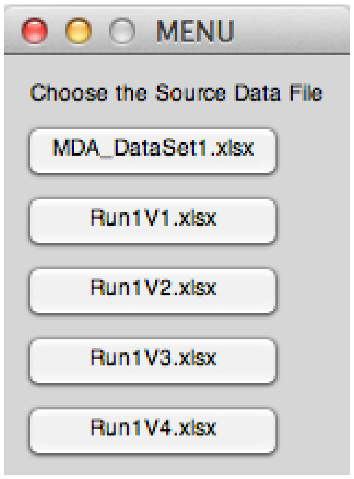


Figure 10. The MATLAB graphical user interface with push-button input.

Table 7. Synthetic and real-world analyses model parameters.

Parameter	Value for Synthetic and AIS Real-World Scenarios	Value for GPS Real-World Scenario
Time Window Length	12 Minutes	1 Minute
Maximum Heading Difference Between Contacts	39°	10°
Maximum Distance Between Contacts	8 Nautical Miles	0.5 Nautical Miles
Maximum Speed Difference Between Contacts	16 Knots	5 Knots
Moving Cluster Threshold Value	60%	25%

B. VERIFICATION USING SYNTHETIC DATA

The simulation is run using data that was purposefully created to test the functionality of the STC algorithm. Specifically, the synthetic scenario provides proof-of-concept that the algorithm can properly evaluate the user-selected values from Table 7 and that the text output and the interactive aspect of the visual output are operable. The data simulation involves a dataset of three hundred vessel position reports taken over a period of seventy-two minutes, or six time steps. The setup and results of the simulation are presented below.

1. Simulation Setup

In the simulation, two separate scenarios are executed. Scenario one, which occurs on the top half of Figures 11–16, consists of two contacts and is designed to test maximum heading difference between contacts and maximum distance between contacts. Scenario two, which occurs on the lower half of Figures 11–16, begins with two groups of five contacts each, and is designed to test maximum speed difference between the contacts, maximum distance between contacts, and the operability of the moving cluster threshold value. The narrative steps for both scenarios, as well as the expectation for kinematic clustering, are detailed in Table 8.

Table 8. Summary of the vessel interactions for the simulation scenarios.

Time-Step	Scenario One	Expectations	Scenario Two	Expectations
1	Distance: 9 nm Heading Diff: 38°	No Cluster	Distance: 9 nm Speed Diff: 17kts	2 Clusters
2	Distance: 7.8 nm Heading Diff: 38°	Cluster	Distance: 7.3 nm Speed Diff: 17kts	2 Clusters
3	Distance: 6.6 nm Heading Diff: 0°	Cluster	Distance: 5.6 nm Speed Diff: 15kts	1 Cluster
4	Distance: 6.6 nm Heading Diff: 0°	Cluster	Distance: 7.1 nm Speed Diff: 15kts	1 Cluster
5	Distance: 6.6 nm Heading Diff: 40°	No Cluster	Distance: 5.6 nm Speed Diff: 15kts Lose 4 contacts	1 Cluster
6	Distance: 7.8 nm Heading Diff: 40°	No Cluster	Distance: 4.1 nm Speed Diff: 15kts Lose 3 contacts	1 Cluster

In scenario one, the contacts begin nine nautical miles apart with a course heading difference of 38° at time-step one. The contacts are traveling at a constant speed of 18 knots and at time-step two are 7.8 nautical miles apart with the same course heading difference. At this point, because they now fall inside the eight nautical mile threshold and have less than a 39° heading difference, the two contacts should be classified as a kinematic cluster. At time-step three, the contacts are 6.6 nautical miles apart and have both turned to course 090. With a 0° course heading difference, the contacts should again be stored as a kinematic cluster. The contacts maintain course and speed through time-step four, and should, therefore, be kinematically clustered. At time-step five, the contacts turn on outbound courses with a heading difference of 40°. Although they are still only 6.6 nautical miles apart, the contacts' course heading difference has now exceeded the defined threshold, and they should no longer be considered a kinematic cluster. They continue on their outbound courses at time-step six and should not be kinematically clustered. The contacts in scenario one should be considered a moving cluster beginning at time-step 2 and ending at time-step four.

In scenario two, two groups of five contacts each begin nine nautical miles apart traveling on identical courses with a speed difference between the groups of 17 knots. Due to the speed difference and distance between them, the groups should be classified as two separate kinematic clusters, as the five contacts in each group are traveling at the same speed. At time-step two, the two groups are inside the eight nautical mile threshold, but still maintain a 17-knot speed difference and, therefore, should be clustered separately. At this point, each of the groups should be classified as a moving cluster that begins at time-step one. The two groups of contacts shift speed at time-step three and settle to a 15-knot speed difference. Now, inside the speed difference threshold of 16 knots, the two groups should be kinematically clustered as one group of ten contacts. At time-step four, the contacts shift speed again but maintain their speed difference at 15 knots. They should again be clustered as one group of ten contacts and now considered a moving cluster that begins at time-step three. At time-step five, the contacts maintain their speed, but four of the ten contacts fail to transmit a position report in order to test the moving cluster threshold value of 60%. The remaining contacts should continue to be kinematically clustered, and because 60% of the contacts of the moving cluster remain, they should be considered a moving cluster. At time-step six, the contacts maintain speed, but three of the remaining six contacts do not have associated position reports. The remaining three contacts should be kinematically clustered, but because only 50% of the six contacts remain, the moving cluster should be reported to end at time-step five.

2. Kinematic Clustering Results

The simulation scenarios have been described in detail above, and expectations for kinematic clustering and moving cluster analysis have been set. In the discussion to follow, the kinematic results at each time-step and the spatiotemporal outcome will be detailed.

The visual representation of the kinematic clusters at time-step one is presented in Figure 11. As expected, the only kinematic clusters displayed are those from scenario two, each consisting of five contacts, colored cyan and green and labeled c_1 and c_2, respectively, in Figure 11. The black dots represent other contacts in the input dataset that are not kinematically clustered in any cluster.

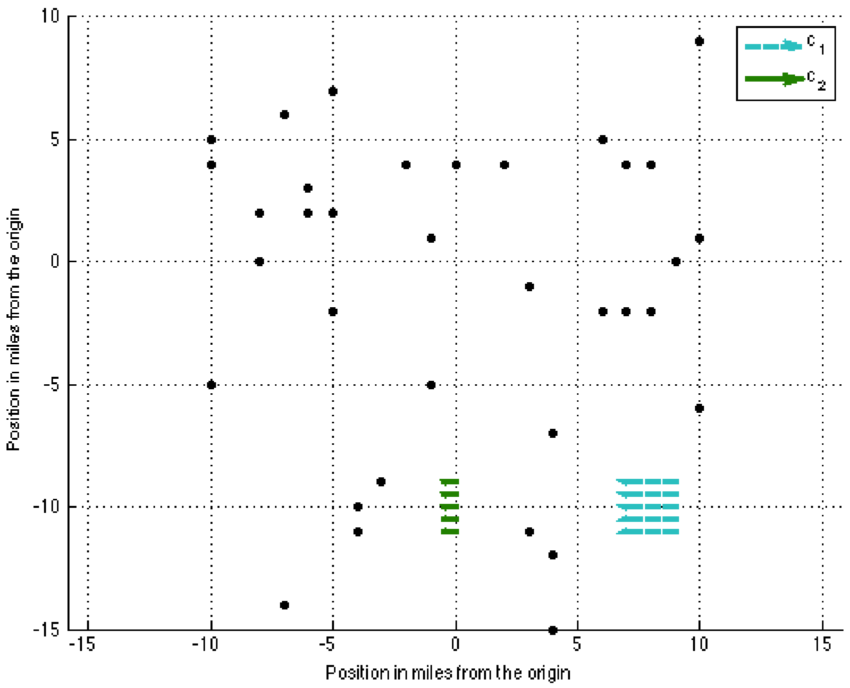


Figure 11. The kinematic clustering results for time-step one, which reveal two clusters, each containing five contacts, are in accordance with the expected outcome.

The kinematic clustering results for time-step two are shown in Figure 12. In addition to the kinematic clusters for scenario two, colored green and cyan and labeled c_1 and c_3, respectively, the two contacts from scenario one are now kinematically clustered, colored red, and labeled c_2.

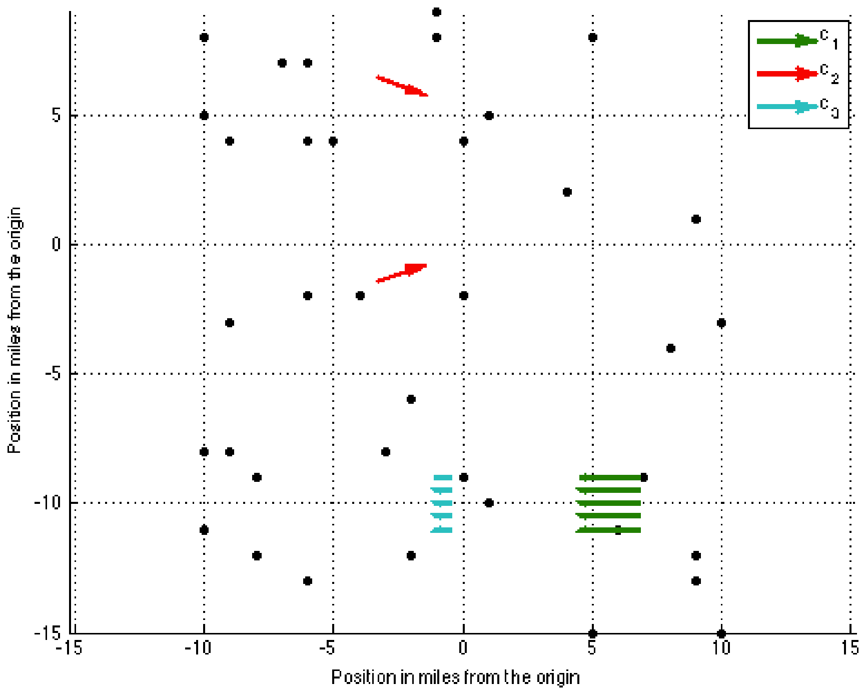


Figure 12. The kinematic clustering results for time-step two, which reveal three clusters, cyan and green each containing five contacts, and red with two contacts, are in accordance with the expected outcome.

At time-step three, the expectation was for the kinematic cluster from scenario one to continue, but for the two kinematic clusters from scenario two to be combined into one. The results, illustrated in Figure 13, are as expected. The scenario one cluster is colored purple and labeled c_2, and the new larger kinematic cluster from scenario two is colored cyan and labeled c_1.

The visual representation of the kinematic clusters at time-step four is presented in Figure 14. In time-step four, all contacts maintained course and speed and were expected to be kinematically clustered as in time-step three.

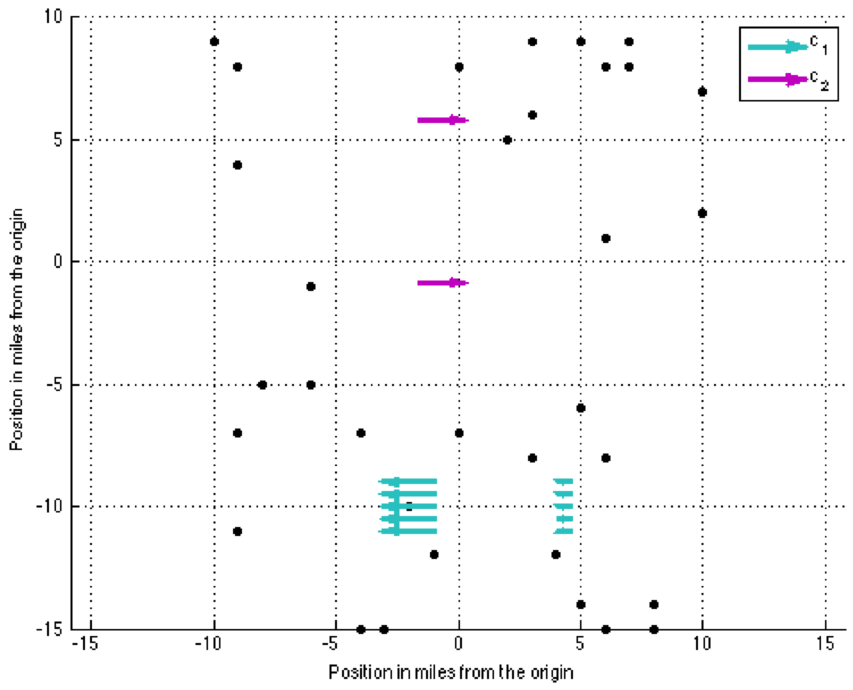


Figure 13. The kinematic clustering results for time-step three contain two clusters: green, which is the combination of the two five-contact clusters from time-step two, and red, with two contacts. The clusters are as expected.

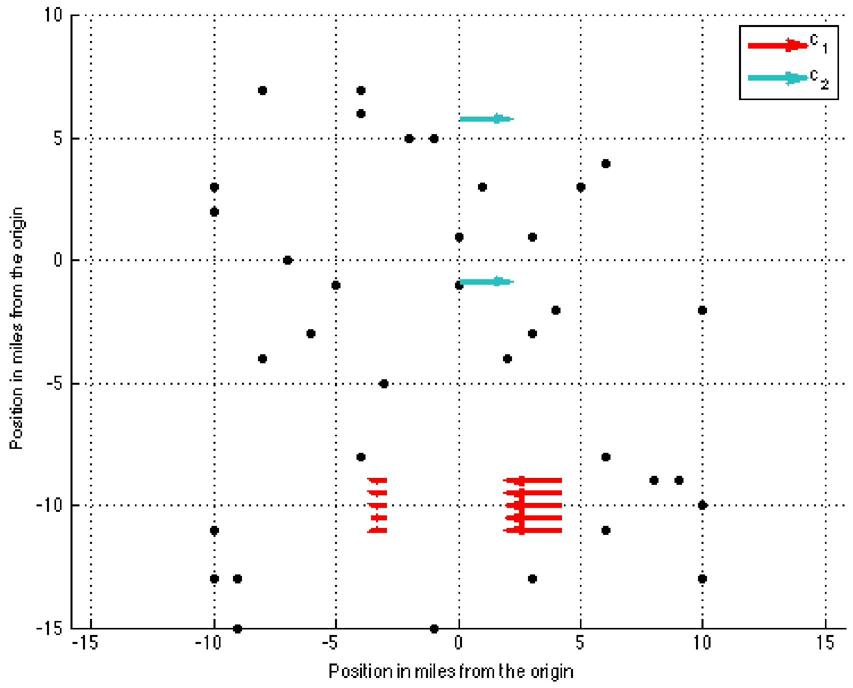


Figure 14. The kinematic clustering results for time-step four, which reveal the same two clusters as in time-step three, are as expected.

At time-step five, we do not expect the contacts in scenario one to be kinematically clustered due to exceeding the maximum heading difference threshold. As illustrated in Figure 15, the cluster from scenario one no longer appears. Four contacts from scenario two did not transmit position reports, but the remaining six should still be kinematically clustered together. In Figure 15, the cluster from scenario two is colored red and labeled c_1.

The contacts in scenario one continue to exceed the maximum heading difference threshold at time-step six and should not be kinematically clustered together. Of the remaining six contacts in scenario two, three fail to transmit a position report. The remaining three contacts should still be kinematically clustered and are illustrated in red and labeled c_1 in Figure 16.

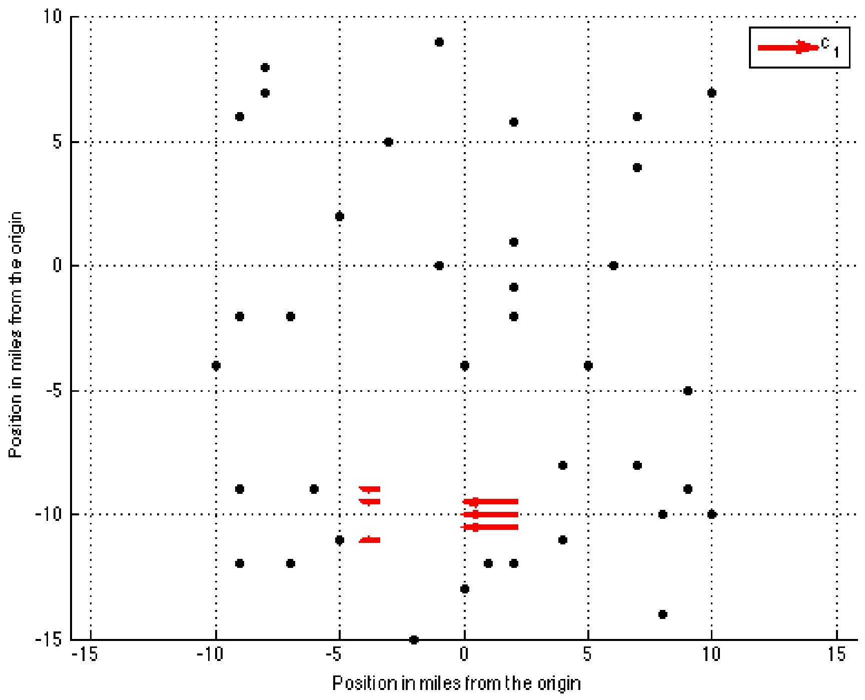


Figure 15. The kinematic clustering result for time-step five contains one cluster, which contains the remaining six contacts from the larger ten-contact cluster in time-step four, is tracked as expected because of the moving cluster threshold value definition of 60%.

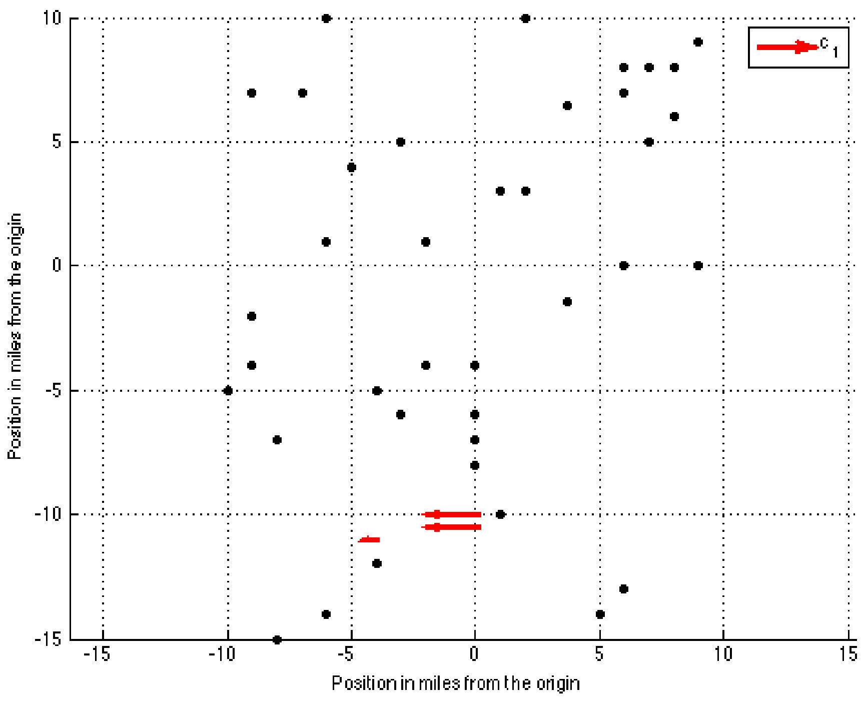
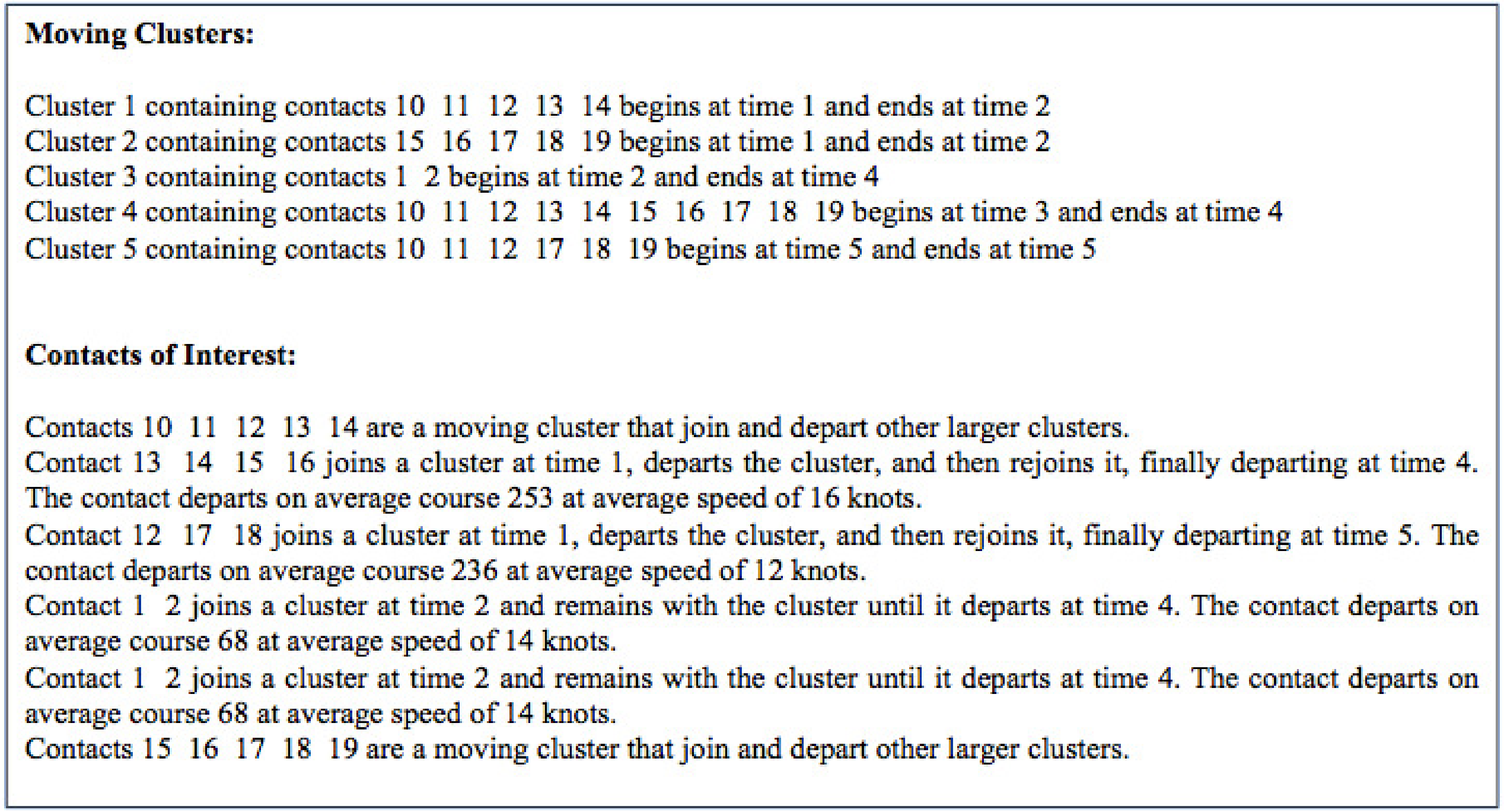


Figure 16. The kinematic clustering result for time-step six contains one cluster, which contains the remaining three contacts from the larger six-contact cluster in time-step five, is tracked as expected as a kinematic cluster.

3. Spatiotemporal Clustering Results

The kinematic clustering results were all as expected from the description of the synthetic simulation scenarios. The STC text results are given in Figure 17. As expected, the two groups of contacts in scenario two are considered moving clusters beginning at time-step one and ending at time-step two when the two combine to form one larger cluster. The memberships of the clusters are detailed as cluster 1 and cluster 2. In Figure 17, cluster 3 contains the membership of the contacts in scenario one. They begin as a moving cluster at time-step two and end at time-step four, as expected. Cluster 4, which is the combination of clusters 1 and 2, begins at time-step three and ends at time-step four, because of the departure of four contacts in time-step five. Cluster five, the six contacts that remain from the large group, is described to begin and end at time-step five. This occurs because the group is still considered a moving cluster from time-step four but does not continue into time-step six due to the loss of three of the remaining six contacts.



Moving Clusters:

Cluster 1 containing contacts 10 11 12 13 14 begins at time 1 and ends at time 2
Cluster 2 containing contacts 15 16 17 18 19 begins at time 1 and ends at time 2
Cluster 3 containing contacts 1 2 begins at time 2 and ends at time 4
Cluster 4 containing contacts 10 11 12 13 14 15 16 17 18 19 begins at time 3 and ends at time 4
Cluster 5 containing contacts 10 11 12 17 18 19 begins at time 5 and ends at time 5

Contacts of Interest:

Contacts 10 11 12 13 14 are a moving cluster that join and depart other larger clusters.
Contact 13 14 15 16 joins a cluster at time 1, departs the cluster, and then rejoins it, finally departing at time 4. The contact departs on average course 253 at average speed of 16 knots.
Contact 12 17 18 joins a cluster at time 1, departs the cluster, and then rejoins it, finally departing at time 5. The contact departs on average course 236 at average speed of 12 knots.
Contact 1 2 joins a cluster at time 2 and remains with the cluster until it departs at time 4. The contact departs on average course 68 at average speed of 14 knots.
Contact 1 2 joins a cluster at time 2 and remains with the cluster until it departs at time 4. The contact departs on average course 68 at average speed of 14 knots.
Contacts 15 16 17 18 19 are a moving cluster that join and depart other larger clusters.

Figure 17. The simulation scenario STC text output detailing five moving clusters and multiple contacts of interest.

The visual representation of the STC results is given in Figure 18. According to the text output in Figure 17, there should be five moving clusters displayed. Upon inspection of Figure 18 it is found that there are indeed five clusters, four are red lines. The fifth moving cluster is a single cyan mark on the plot immediately above the lower-center moving cluster. The moving clusters have been enlarged in Figures 19 and 20 in order to further investigate each moving cluster.

Cluster 3, which contains contacts 1 and 2, has been enlarged in Figure 19. The interactive display of MATLAB has been used to select the middle clustering point, and it is shown that the point is a point in Cluster 3 at time-step three. The line and all marks on it are in red, indicating that the moving cluster maintains 100% membership during its life, as expected. The straight-line nature of the moving cluster track in Figure 19 is created by averaging the position, course and speed of contacts 1 and 2.

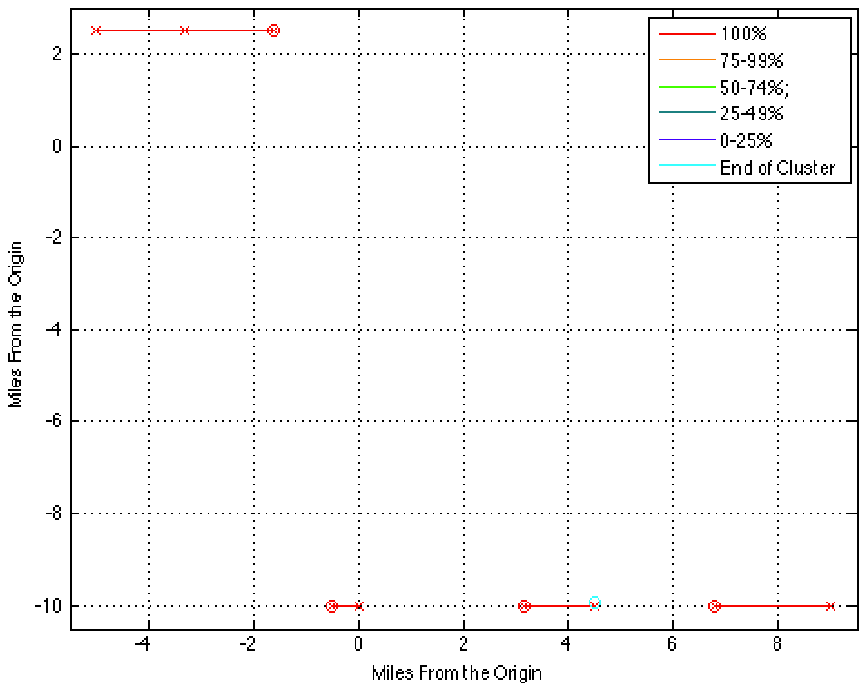


Figure 18. A broad view of the visual representation of the five moving clusters determined in the synthetic scenario, which is as expected.

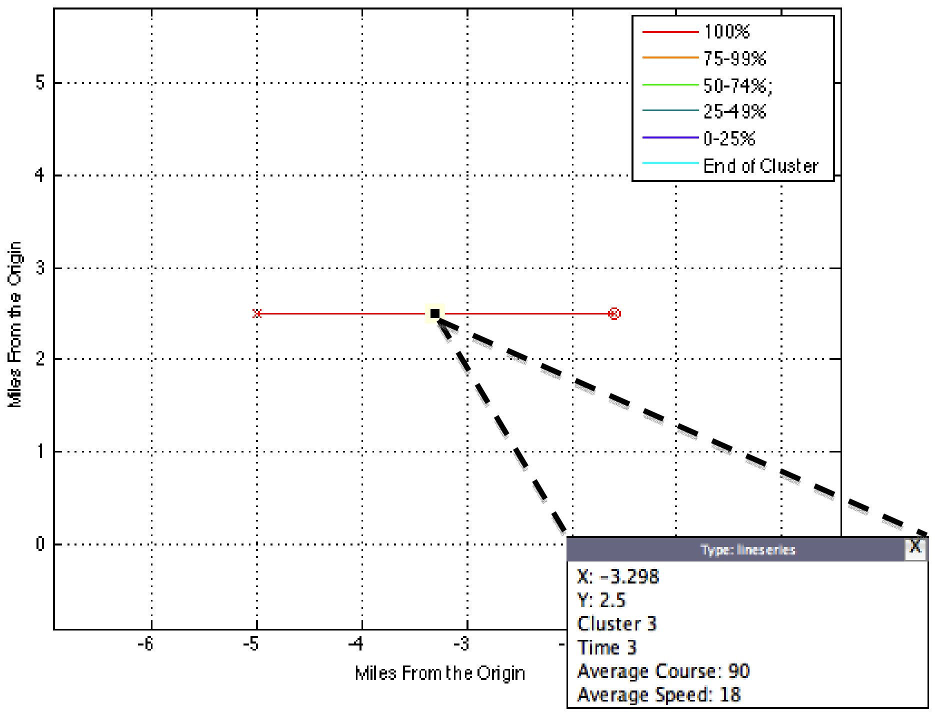


Figure 19. An enlarged view of the visual representation of moving cluster 3, which contains contacts 1 and 2 in the synthetic scenario, exists between time-steps 2 and 4, and is as expected.

The other four moving clusters have been enlarged in Figure 20. The lines on the left and right correspond to Clusters 1 and 2 in Figure 17. Through the interactive feature of the plot it can be seen that both clusters begin at time-step one and end at time-step two when they combine into one larger moving cluster, represented by the center red line in Figure 20. Both clusters have red lines and marks associated with them, indicating 100% membership is maintained for both time steps of their lives. The middle cluster is representative of Cluster 4 from Figure 17, which is the combination of the two smaller moving clusters. Although one mark of the cluster is hidden by the MATLAB plot marker, all marks and the line of the cluster are red, indicating 100% membership during its two-time-step life, which begins at time-step three and ends at time-step four. The MATLAB interactive plot marker is marking Cluster 5 from Figure 17, which begins and ends at time-step five after four of the ten contacts depart the large moving cluster. Under the MATLAB interactive marker the mark is cyan in color, indicating the end of the moving cluster's life.

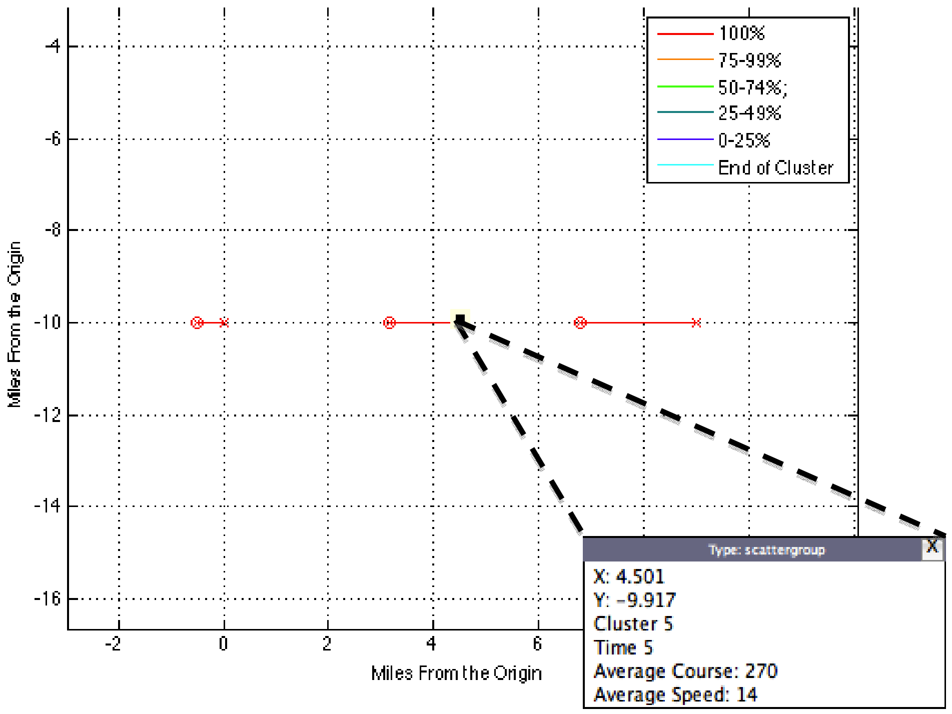


Figure 20. An enlarged view of the visual representation of clusters 1, 2, 4, and 5 in the synthetic scenario. Clusters 1 and 2 combine at time-step 3 to form cluster 4, and cluster 5 is tracked during time-step five as the remaining six contacts of cluster 4.

C. VALIDATION USING REAL-WORLD DATA

After successfully evaluating the synthetic data scenarios, the STC algorithm was tested on two real-world datasets. For the first analysis, the real-world data source is worldwide AIS vessel position reports for the twenty-four hour period beginning at 0001, 10 January 2012. The water space of interest for the analysis was chosen to be the Strait of Malacca. The Strait of Malacca, which is the main shipping channel between the Indian Ocean and the Pacific Ocean, is considered one of the most important shipping lanes in the world, with more than 60,000 vessels passing through it each year [26].

The purpose of the real-world data analysis is to illustrate the STC algorithm's ability to process a robust and relatively large set of data. On 10 January 2012 there were slightly more than 1.5 million worldwide AIS vessel position reports, approximately five thousand of which occurred in and around the Strait of Malacca. The complete AIS position report picture is shown in Figure 21. Each circle on the plot represents an individual AIS position report during the time period of interest.

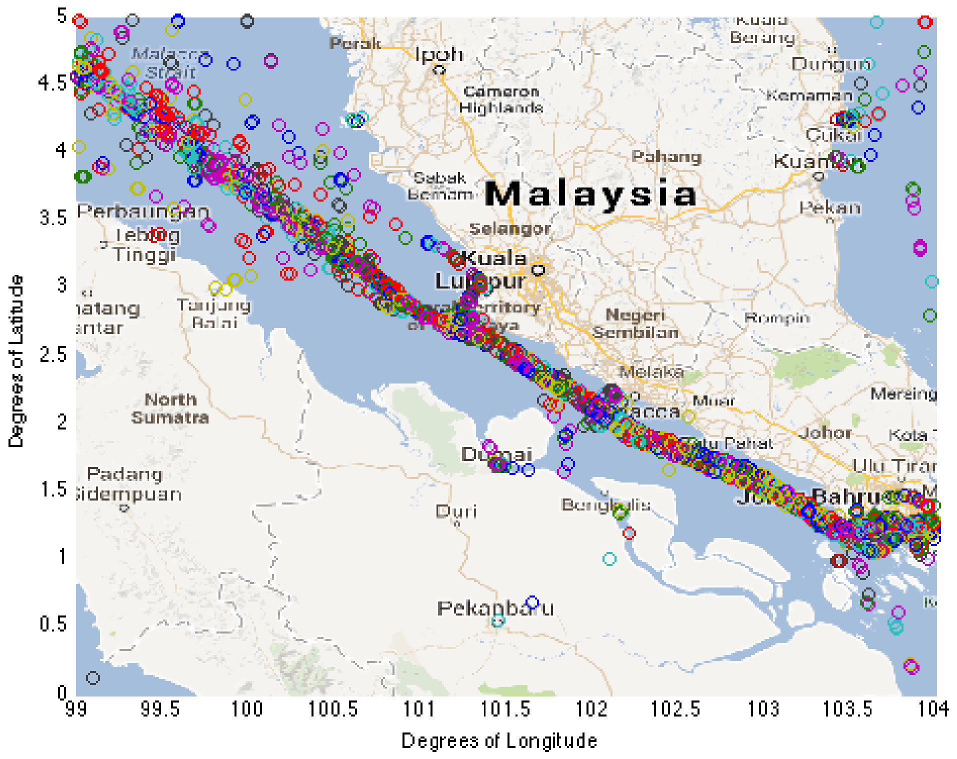


Figure 21. Visual representation of the Strait of Malacca AIS position report dataset for 10 January 2012.

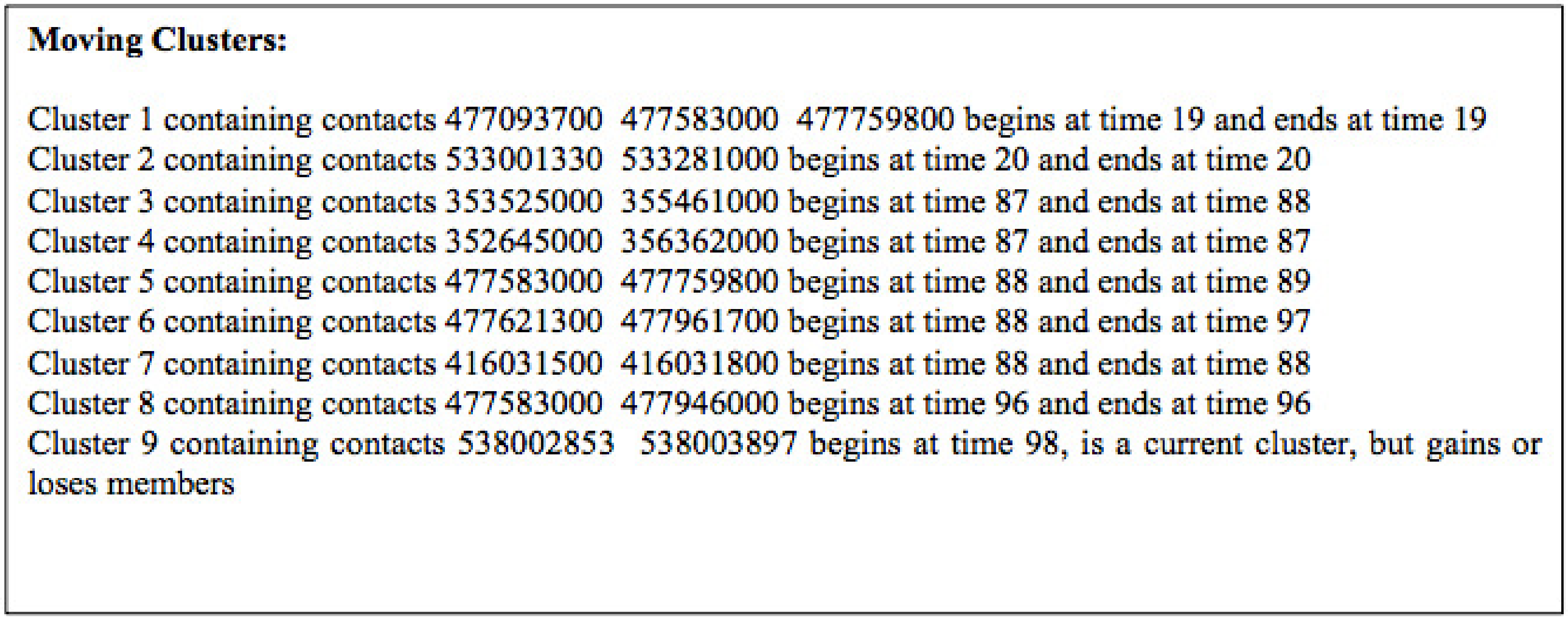
1. AIS Real-world Data Concerning the Strait of Malacca

To set up the scenario, we present a hypothetical report of intelligence that the motor vessel (M/V) Mairini, with MMSI number 538003897 has conducted an illicit transfer of cargo or persons on 10 January 2012 in or around the Strait of Malacca. Theater strategic decision makers need to determine if M/V Mairini interacted with or traveled with any other ships for any period of time during the day in question. The worldwide AIS dataset for 10 January is provided to the STC algorithm, and the results are detailed below.

2. AIS Real-world Data Results

Twelve moving clusters are reported as a result of the STC algorithm analysis using the threshold values defined in Table 7. The text output results of the Strait of Malacca analysis are presented in Figure 22. Through examination of the text output, we can determine that the VOI, MMSI 538003897, appears to interact with another vessel on the day in question in the Strait of Malacca. By converting the time steps to time of day, we can determine that M/V Mairini began interacting with another vessel around 1936 UTC on 10 January 2012. The STC algorithm text output reports that the VOI and the vessel it is clustered with, MMSI 538002853, are a current cluster as well. The latest time-step of data that contains kinematic clusters is time-step 116, which converts to 2312 UTC. An overview of the visual output is presented in Figure 23.

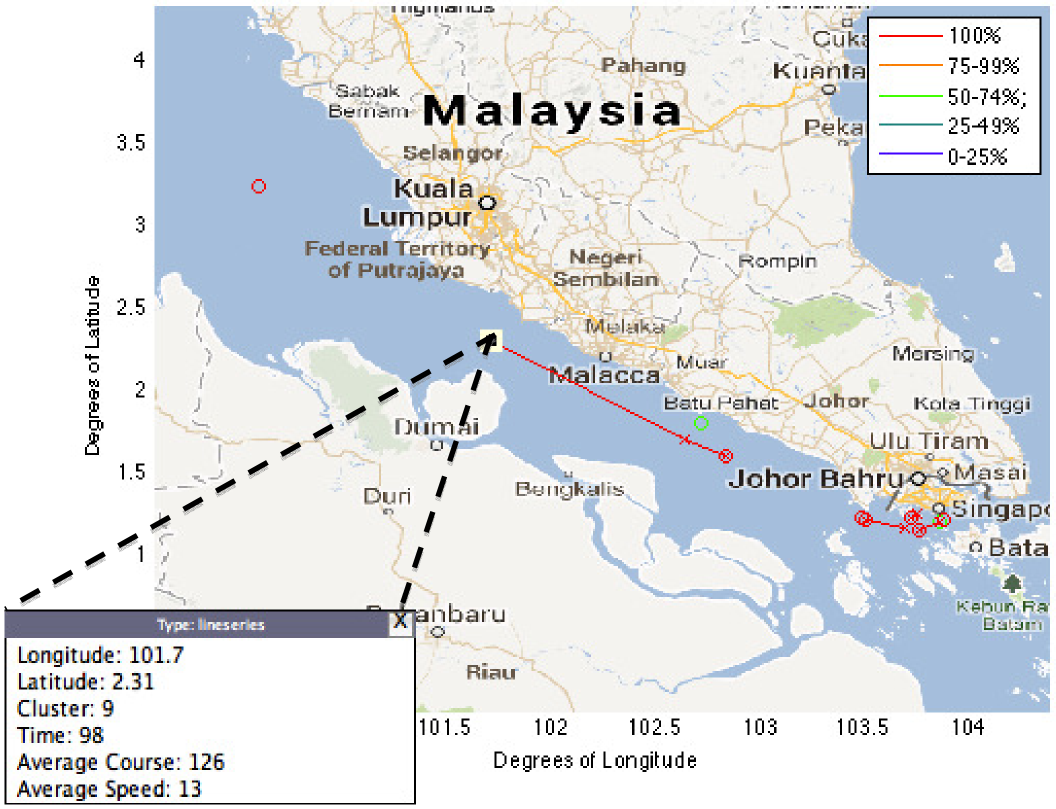
The underlay of the Google map on the visual output provides situational awareness to the user. The MATLAB interactive capability that was highlighted in the synthetic simulation provides further insight into the situation. The moving cluster in question, cluster 9, has been selected in Figure 23, and its attributes are displayed. The cluster was moving on an average course of 126° at a speed of 13 knots at time-step 98, and when the two vessels are last clustered at time-step 116, they are moving on an average course of 114° at a speed of 14 knots. From the visual representation in Figure 23, it appears that the moving cluster is on a standard transit course through the Strait of Malacca. While the results of the real-world data analysis are not conclusive evidence that M/V Mairini was involved in illicit behavior, they do offer enough insight as to the vessel's interactions on the day in question for an analyst's further scrutiny.



```
Moving Clusters:

Cluster 1 containing contacts 477093700  477583000  477759800 begins at time 19 and ends at time 19
Cluster 2 containing contacts 533001330  533281000 begins at time 20 and ends at time 20
Cluster 3 containing contacts 353525000  355461000 begins at time 87 and ends at time 88
Cluster 4 containing contacts 352645000  356362000 begins at time 87 and ends at time 87
Cluster 5 containing contacts 477583000  477759800 begins at time 88 and ends at time 89
Cluster 6 containing contacts 477621300  477961700 begins at time 88 and ends at time 97
Cluster 7 containing contacts 416031500  416031800 begins at time 88 and ends at time 88
Cluster 8 containing contacts 477583000  477946000 begins at time 96 and ends at time 96
Cluster 9 containing contacts 538002853  538003897 begins at time 98, is a current cluster, but gains or
loses members
```

Figure 22. The STC algorithm text output for the AIS real-world data analysis of the Strait of Malacca on 10 January 2012.



Type: lineseries
Longitude: 101.7
Latitude: 2.31
Cluster: 9
Time: 98
Average Course: 126
Average Speed: 13

Figure 23. STC algorithm visual output for a AIS real-world data analysis of the Strait of Malacca on 10 January 2012. Moving cluster 9 containing two contacts travels on average heading 126 at an average speed of 13 knots.

3. GNSS Real-world Data Concerning Vehicles Imitating Small Boats

The second real-world data analysis is performed on GPS data gathered over a twenty-three minute period from 1514 to 1537 UTC on 24 April 2013 via GNSS transmitters mounted on six vehicles driving on Route 3 in Massachusetts. The vehicles operate in a manner to simulate small boats traveling together. The drive-plan for the vehicles dictated three of the them to form a convoy and maintain speed and distance for the span of the exercise. The other three vehicles acted as confuser vehicles and moved in and out of the convoy. One of the convoy vehicles was unable to report heading data and was, therefore, excluded from STC algorithm analysis. The analysis from the STC algorithm reveals that multiple moving clusters are tracked over the twenty-three minute period, which is in keeping with the drive-plan, which is summarized in Table 9.

Table 9. GNSS real-world data analysis vehicle drive-plan.

Vehicle	Drive Plan
1	Confuser: drive alongside/lead/trail/enter/exit convoy, pass other confusers
2	Confuser: drive alongside/lead/trail/enter/exit convoy
3	Convoy lead: drive at/near speed limit in right lane
4	Convoy middle: do not change convoy order, vary follow spacing, allow confusers to enter/exit convoy
5	Confuser: drive alongside/lead/trail/enter/exit convoy
6	Convoy tail: do not change convoy order, vary follow spacing, allow confusers to enter/exit convoy

4. GNSS Real-world Data Results

According to the drive-plan, vehicles 3, 4, and 6 form the convoy and should regularly be clustered as a moving cluster over the span of the twenty-three minute dataset. Vehicle 3 did not transmit heading data, so it was not analyzed by the STC algorithm. The remaining vehicles should move in and out of the convoy and form various moving clusters during the run. From the visual representation in Figure 24 and the text output in Figure 25, it is evident that several moving clusters are tracked in the analysis of the dataset. Vehicles 4 and 6, which form the convoy, are not explicitly clustered together for the length of the analysis but are detailed in the "Contacts of Interest" section as a moving cluster that joins and departs other clusters. This designation provides evidence that contacts 4 and 6 move together for much of the twenty-three minute run as directed in the drive-plan. With user thresholds set as previously detailed, thirteen moving clusters are identified during the analysis. The confuser vehicles moving in and out of the convoy account for a majority of these cluster changes. In Figure 24 moving cluster 9 consisting of vehicles 1, 2, and 6 is identified.

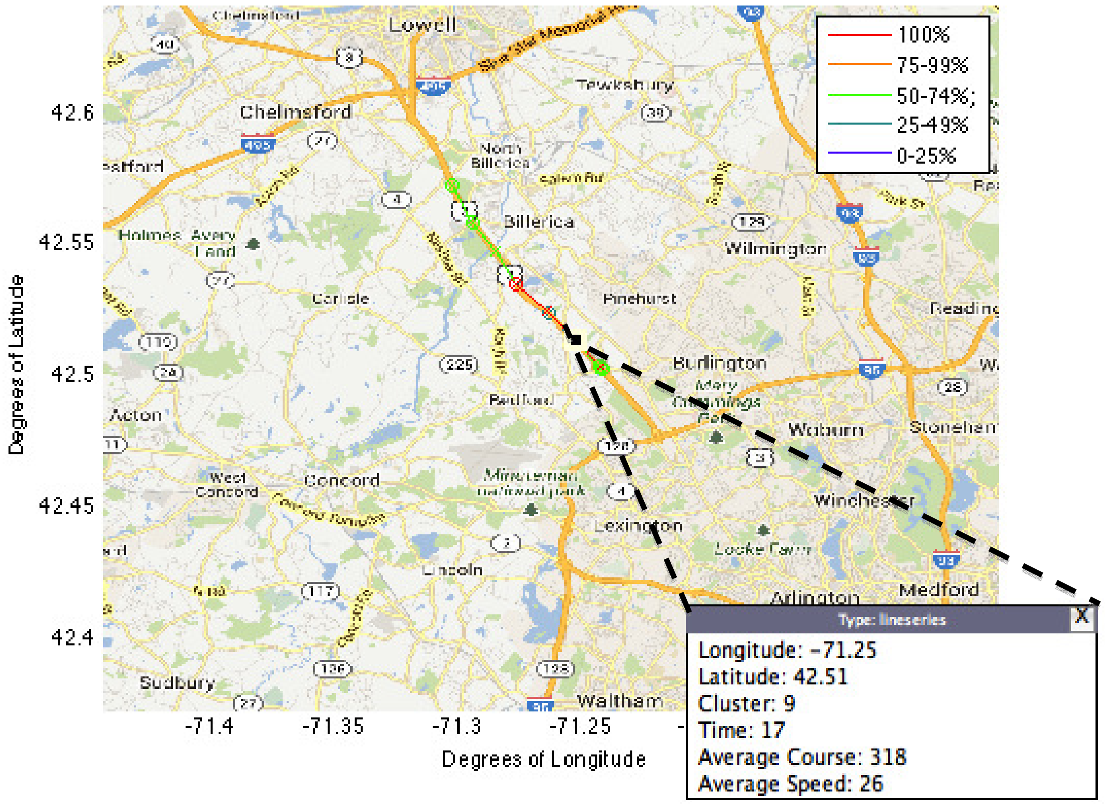
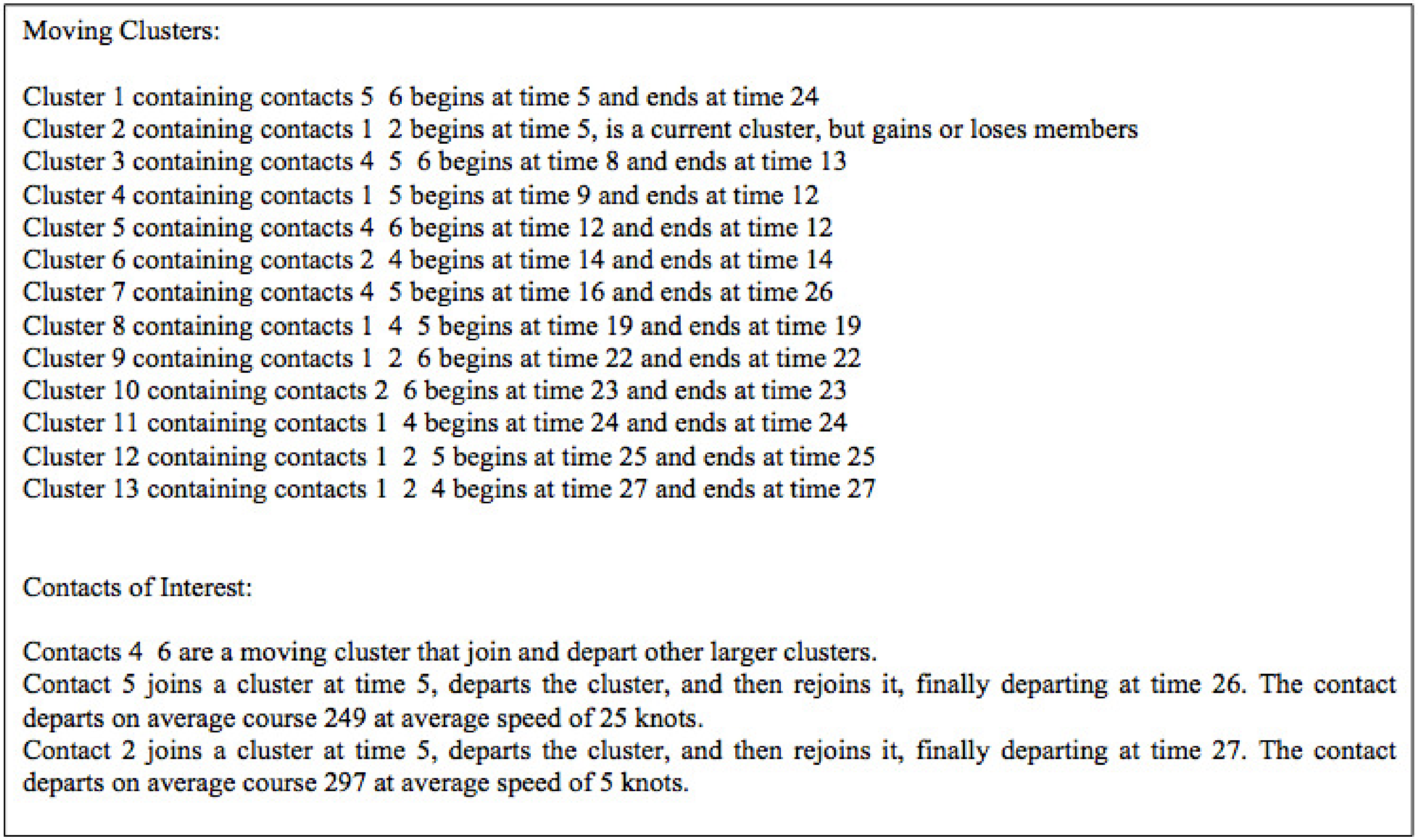


Figure 24. STC algorithm visual output for a GNSS real-world data analysis of vehicles imitating small boats on 24 April 2013.

In summary, the implementation of the STC algorithm in the MATLAB programming environment was detailed in this chapter. A synthetic scenario was used to verify the algorithm's filtering and clustering logic and user-provided threshold limits. Two real-world datasets were used to validate the STC algorithm's performance. The first used AIS data and a hypothetical scenario to analyze the interactions of a commercial VOI. The second used GNSS data consisting of vehicles simulating small boats operating at higher speeds and closer distances than the commercial AIS scenario.



```
Moving Clusters:

Cluster 1 containing contacts 5  6 begins at time 5 and ends at time 24
Cluster 2 containing contacts 1  2 begins at time 5, is a current cluster, but gains or loses members
Cluster 3 containing contacts 4  5  6 begins at time 8 and ends at time 13
Cluster 4 containing contacts 1  5 begins at time 9 and ends at time 12
Cluster 5 containing contacts 4  6 begins at time 12 and ends at time 12
Cluster 6 containing contacts 2  4 begins at time 14 and ends at time 14
Cluster 7 containing contacts 4  5 begins at time 16 and ends at time 26
Cluster 8 containing contacts 1  4  5 begins at time 19 and ends at time 19
Cluster 9 containing contacts 1  2  6 begins at time 22 and ends at time 22
Cluster 10 containing contacts 2  6 begins at time 23 and ends at time 23
Cluster 11 containing contacts 1  4 begins at time 24 and ends at time 24
Cluster 12 containing contacts 1  2  5 begins at time 25 and ends at time 25
Cluster 13 containing contacts 1  2  4 begins at time 27 and ends at time 27


Contacts of Interest:

Contacts 4  6 are a moving cluster that join and depart other larger clusters.
Contact 5 joins a cluster at time 5, departs the cluster, and then rejoins it, finally departing at time 26. The contact
departs on average course 249 at average speed of 25 knots.
Contact 2 joins a cluster at time 5, departs the cluster, and then rejoins it, finally departing at time 27. The contact
departs on average course 297 at average speed of 5 knots.
```

Figure 25. STC algorithm text output for GNSS real-world data analysis of vehicles imitating small boats on 24 April 2013.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS

The focus of this thesis was to develop a STC algorithm to aid in the Navy's mission of MDA. Specifically, the STC algorithm was designed to autonomously analyze vessel interactions to gain behavioral knowledge of a water space. An extension of the urban combat environment direction and displacement directivity algorithms was proposed as a method of classifying paralleling and following movement which may be exhibited by vessels involved in illicit activity.

The general framework of STC in this thesis was to form clusters kinematically and then refine them temporally. Clusters were formed through proximity filtering with k-means clustering, kinematically refined by determination of the similarity of vessel course, speed, and distance between vessels, and further refined through temporal analysis. This methodology was deemed to effectively provide a refined kinematic clustering result that carefully balanced user input with mathematical reasoning. The kinematic clusters were then temporally analyzed to determine which clusters were moving in time and which were not. Spatiotemporal results were then processed to produce usable text and visual outputs that result in better understanding of vessel interactions and behavioral insight in a water space.

A. SIGNIFICANT RESULTS

The work presented in this thesis provides three contributions to the MDA problem set. First an STC scheme was developed and applied to the maritime domain to identify interactions between vessels at sea in order to gain behavioral knowledge of paralleling and following movement in a water space. The scheme was validated using multiple data sources that provided a real-world hypothetical scenario in the Strait of Malacca and a simulated movement of small boats.

Second, three filters were designed in support of the operation of the STC algorithm. Location filtering enabled a large input dataset to be focused onto a specific water space of interest. The proximity filter spatially refined the input dataset by grouping contacts via k-means clustering centroid assignment. Kinematic filtering then determined the similarity of vessel course and speed within each k-means cluster and further spatially refined the clustering process by comparing the distance between vessels.

Finally, the STC algorithm provided those interested in national or theater security decision making the opportunity to quantify the (membership) continuity of a moving cluster. A group of vessels deemed to be exhibiting paralleling or following behavior by the algorithm were tracked until the continuity fell below the user-defined threshold of interest. The opportunity to quantify the continuity allows the user to characterize the degree to which a moving cluster may change.

B. FUTURE WORK

In this thesis the focus of the STC algorithm was on a water space of interest. Use of the VOI identifier as input to determine the water space of interest during the time of interest would improve the algorithm's performance. Future work should extend the algorithm to include VOI identity if available.

An extension of this work would be to focus on countering the knowledgeable enemy. In this thesis the assumption was made that vessels will not attempt to prevent detection of their behaviors. If, for instance, vessels rendezvoused with other vessels in a way that inhibited the algorithm from detection as a moving cluster, the advantage provided by the STC algorithm would be nullified. Perhaps the vessels would meet via a head-on scenario or would steer headings that were always greater than 90° in difference in order to avoid detection. Expanding the current algorithm to detect a head-on meeting scenario would improve its performance.

The STC algorithm presented in this thesis does not provide track estimation functions. In future work, when a position report is missing or unreadable, the vessel's movement could be tracked through the use of a Kalman filter or a similar method. This would prevent negative effects on the formation or tracking of a moving cluster and would allow for greater insight and analysis of the water space.

The STC algorithm’s scope could be expanded to include more than commercial vessels. Given the speed that a small boat swarm might attack with, early detection of their formation is an important task. An extension of this thesis work could be applied to the detection of such a swarm by applying short time windows and small thresholds for distance, heading, and speed differences between contacts. Although the second real-world dataset was meant to mimic small boats, conducting the STC analysis on actual small boat data would be beneficial.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX

The MATLAB code used to implement the spatiotemporal clustering analysis is provided in the appendix.

```
function Thesis
% Thesis    Top-Level Function Call.
%   Thesis is the function call to access spatiotemporal
algorithms that support maritime domain awareness. The user will
select whether their input data is “Synthetic” or “Real-World,”
and the program will determine which set of functions to use for
analysis.
%   Created by LT Kristofer Tester, USN, April 2013

% Function Input:     None
% Function Output:    Proper file path alignment

clear all
close all
clc
t = cputime;

% Display a menu offering the user choice of data type
z = menu(‘Select Source Data Type’,’Synthetic’,’Real-World’);

% Process user selection and choose next function call. Matlab
directory changes to reflect user selection, and to keep data
files separated and easily organzied.
if z == 1
    oldFolder = cd(‘Thesis Synthetic’);
    Synthetic;
    cd(oldFolder);
else
    oldFolder = cd(‘Thesis RealWorld’);
    RealWorld;
    cd(oldFolder);
end

% Display the time, in seconds, the program takes to run in its
entirety
t = cputime
```

```
function RealWorld

% REALWORLD    Top-level function call for real-world data
analysis.
%   RealWorld is the top-level function call for real-world AIS
data analysis. This function begins with data pre-processing and
filtering, and includes all functions required for spatiotemporal
analysis of the waterspace. Created by LT Kristofer Tester, USN,
April 2013

% Call function to read data input and perform preprocessing
duties
[A,default,climit,spdlimit,dlimit,spdinput,storage] =...
    InputandPreprocessing;

% Loop through the data by time-step. At each time-step, arrange
a submatrix, B, of the A matrix, such that all members of B have
the same time-step assignment. Proximity filter and kinematically
cluster the contacts in the B matrix using the KDVC function and
return the results in clusters. Store clusters in the cell array,
compold.
[compold,Data,maxtime,clusters1] =
ProxKin(A,default,climit,spdlimit,...
    dlimit,spdinput,storage)

% Call function Temporal to perform temporal analysis of the
kinematic time step snapshots
[MovingClusters,FuseClusters,DiffClusters,CurrentClusters,timecou
nt] =...
    Temporal(compold,storage)

% Build the text output by comparing each cell of MovingClusters.
If the cell is found elsewhere, determine what time-step it
appears, and what time-step it disappears. Output this
information, along with the membership details of the moving
cluster in a text format.
Postprocessing(MovingClusters,FuseClusters,DiffClusters,CurrentCl
usters,...
    timecount,Data,maxtime,clusters1)
```

```
function [A,default,climit,spdlimit,dlimit,spdinput,storage]...
    = InputandPreprocessing;

% InputandPreprocessing    This function reads the input data
stream and formats it for use by the algorithm.
% Created by LT Kristofer Tester, USN, April 2013

% Display push-button menu for user selection of the input data
file type
a = menu('Select Data Format:','xlsx Format','.mat Format','csv
Format');

% List all files of the type xlsx and .mat in the current
directory and store the filenames in TestList cell array for user
selection
if a == 1;
    TestFiles = dir('*xlsx');
    TestList = {};
    for i = 1:length(TestFiles)
        filename = TestFiles(i).name;
        TestList{i} = filename;
    end
    z = menu('Choose the Source Data File',TestList);
    Data = xlsread(TestList{z});
elseif a == 2;
    TestFiles = dir('*.mat');
    for b = 1:length(TestFiles)
        filename = TestFiles(b).name;
        TestList{b} = filename;
    end
    z = menu('Choose the Source Data File',TestList);
    Data = load(TestList{z});
    cellData = struct2cell(Data);
    matData = cell2mat(cellData);
    Data = matData;
elseif a == 3;
    TestFiles = dir('*csv');
    for j = 1:length(TestFiles)
        filename = TestFiles(j).name;
        TestList{j} = filename;
    end
    z = menu('Choose the Source Data File',TestList);
    Data = csvread(TestList{z});
end

z =...
menu('Choose Global Location or Manually Input Area of Interest
Coordinates',...
'Strait of Malacca','West Coast of Africa','Panama Canal','Bab Al
Mandeb',...
'Manual','Entire Area');
if z == 1;
    locationFiltered = locationFilter(Data,2,102,4,4);
elseif z == 2;
    locationFiltered = locationFilter(Data,20,-18,5,5);
elseif z == 3;
    locationFiltered = locationFilter(Data,7,-80,5,5);
```

```
elseif z == 4;
    locationFiltered = locationFilter(Data,12.5,43.5,5,5);
elseif z == 5;
    handles = guihandles(LocationFilt);
    centlat = get(handles.slider1,'Value');
    centlon = get(handles.slider2,'Value');
    height = get(handles.slider3,'Value');
    width = get(handles.slider4,'Value');
    close(handles.figure1);
    locationFiltered =
locationFilter(Data,centlat,centlon,width,height);
elseif z == 6;
    locationFiltered = Data;
end

% Store the source data in the Data variable and assign variables
for the necessary columns of the Data matrix
Data = locationFiltered;
Lon = Data(:,6);
Lat = Data(:,7);
Cog = Data(:,8);
Sog = Data(:,5);
UTC = Data(:,10);
counter = Data(:,2);

% Create the A matrix - the variables from Data that are
necessary for analysis
A = [Lon Lat Cog Sog counter UTC];

% Format the A matrix using Data_Read
default = menu('Parameter Selection','Default Values',...
    'User Selected Values');
[A] = Data_Read(A,default);

% Add longitude and latitude component of each contact to the A
matrix for use in output formats
A = [A Lon Lat];

% Initialize looping variables and determine the unique MMSI
identifiers that are reported in the A matrix
sizedata = size(A);
n = 1;
k = unique(A(:,7));

% Loop through the unique MMSI identifiers and quiver plot each
contact. This plot represents all of the individual contacts
reported in the data set.
figure();
while n <= length(k)
    j = find(A(:,7)==k(n));
    scatter(A(j,9),A(j,10));
    title('Example Contact Picture')
    xlabel('Degrees of Longitude')
    ylabel('Degrees of Latitude')
    set(gcf,'color','w');
    hold on
    n = n+1;
```

```
end
plot_google_map

% Initialize looping variables
n = 1;

% Create slider and text input GUI for user input
if default == 1;
    climit = 39;
    spdlimit = 16;
    dlimit = 8;
    spdinput = 2;
    storage = 0.6;
elseif default == 2;
    handles = guihandles(SlideFilt);
    climit = get(handles.slider1,'Value');
    spdlimit = get(handles.slider2,'Value');
    dlimit = get(handles.slider3,'Value');
    spdinput = get(handles.slider4,'Value');
    storage = (get(handles.slider5,'Value'))/100;
    close(handles.figure1);
end

% Loop through the A matrix to ensure none of the variables have
a value of exactly zero
while n <= sizedata(1)
    if A(n,1) == 0
        A(n,1) = 0.001;
    else
        A(n,1) = A(n,1);
    end
    if A(n,2) == 0
        A(n,2) = 0.001;
    else
        A(n,2) = A(n,2);
    end
    if A(n,4) == 0
        A(n,4) = 0.001;
    else
        A(n,4) = A(n,4);
    end
    if A(n,3) == 0
        A(n,3) = 0.001;
    else
        A(n,3) = A(n,3);
    end
    if A(n,5) == 0
        A(n,5) = 0.001;
    else
        A(n,5) = A(n,5);
    end
    if A(n,6) <= spdinput;
        A(n,:) = 0;
    else
        A(n,6) = A(n,6);
    end
    n = n+1;
```

```
end

% Assign variables for the columns of the A matrix
x = nonzeros(A(:,1));
y = nonzeros(A(:,2));
dx = nonzeros(A(:,3));
dy = nonzeros(A(:,4));
cse = nonzeros(A(:,5));
spd = nonzeros(A(:,6));
counter = nonzeros(A(:,7));
timestep = nonzeros(A(:,8));
Lon = nonzeros(A(:,9));
Lat = nonzeros(A(:,10));

% Store the variables back into the A matrix
A = [x y dx dy cse spd counter timestep Lon Lat];
```

```
function [locationFiltered] = locationFilter(entry,centralLat,...
    centralLon,width,height)

% LOCATIONFILTER    Filter real-world data by defining a boundary
box.
%   [locationFiltered] =
LOCATIONFILTER(entry,centralLat,centralLon,width,height) filters
the data provided in entry that is located inside a boundary box
defined by the centalLat, centralLon, width, and height. Entry is
a matrix of AIS position reports with latitude and longitude
stored in tenths of degrees. CentralLat/Lon indicate the center
of the area of interest in tenths of degrees of latitude and
longitude. Width and Height indicate teh size of the area of
interest in tenths of degrees of latitude and longitude.
%   Created by LT Ashley McAbee, USN, April 2013

% Define the boundary box
latLowBound = centralLat - height/2;
latUpBound = centralLat + height/2;
longLowBound = centralLon - width/2;
longUpBound = centralLon + width/2;

% Find all datapoints in entry that are positioned inside the
boundary box, and store them in the output variable,
locationFiltered.
locationFilteredIdx = find(entry(:,7) > latLowBound & ...
    entry(:,7) < latUpBound & entry(:,6) > longLowBound &
entry(:,6) <...
    longUpBound);
locationFiltered = entry(locationFilteredIdx,:);
end
```

```
function [A] = Data_Read(A,default)

% DATA_READ    Data conversion file for real-world AIS datasets.
%   [A] = DATA_READ(A) returns a reformatted A matrix after
converting latitude and longitude to x and y coordinates where
the origin is zero degrees latitude and zero degrees longitude.
The function also converts course and speed into dx and dy
velocity components.
% Created by LT Kristofer Tester, USN, April 2013

% Store the incoming data from the A matrix into appropriate
columns to assist in determining the degrees, minutes, and
seconds of latitude and longitude.
Data(:,1) = floor(A(:,2));
Latmin = (A(:,2)-Data(:,1))*60;
Data(:,2) = floor(Latmin);
Latsec = (Latmin-Data(:,2))*60;
Data(:,3) = floor(Latsec);
Data(:,4) = floor(A(:,1));
Lonmin = (A(:,1)-Data(:,4))*60;
Data(:,5) = floor(Lonmin);
Lonsec = (Lonmin-Data(:,5))*60;
Data(:,6) = floor(Lonsec);
Data(:,7) = A(:,3);
Data(:,8) = A(:,4);
Data(:,9) = A(:,6);
counter = A(:,5);

% Organize degrees, minutes, and seconds of latitude and
longitude into components for x and y coordinate plotting.
y1(:,1) = Data(:,1);
y1(:,2) = Data(:,2);
y1(:,3) = Data(:,3);
x1(:,1) = Data(:,4);
x1(:,2) = Data(:,5);
x1(:,3) = Data(:,6);
c(:,1) = Data(:,7);
s(:,1) = Data(:,8);

% Initialize looping variables and loop through each row of the
Data matrix. Determine the degrees of latitude of the contact in
order to determine the conversion matrix, LatDistConv. Multiply
the conversion matrix by the y components from above to determine
a y coordinate on the grid signifying the distance, in nautical
miles, from zero degrees latitude.
n = 1;
while n <= length(Data)
    if Data(n,1) <= 15
        variable = Data(n,1)*((110.649-110.574)/15)+110.574;
        LatDistConv = [variable*0.62 (variable*0.62)/60
((variable*0.62)/60)/60];
        y(n,1) = y1(n,:)*LatDistConv';
        n = n+1;
    elseif Data(n,1) > 15 & Data(n,1) <= 30
        variable = Data(n,1)*((110.852-110.649)/15)+110.649;
        LatDistConv = [variable*0.62 (variable*0.62)/60
((variable*0.62)/60)/60];
```

```
        y(n,1) = y1(n,:)*LatDistConv';
        n = n+1;
    elseif Data(n,1) > 30 & Data(n,1) <= 45
        variable = Data(n,1)*((111.132-110.852)/15)+110.852;
        LatDistConv = [variable*0.62 (variable*0.62)/60
((variable*0.62)/60)/60];
        y(n,1) = y1(n,:)*LatDistConv';
        n = n+1;
    elseif Data(n,1) > 45 & Data(n,1) <= 60
        variable = Data(n,1)*((111.412-111.132)/15)+111.132;
        LatDistConv = [variable*0.62 (variable*0.62)/60
((variable*0.62)/60)/60];
        y(n,1) = y1(n,:)*LatDistConv';
        n = n+1;
    elseif Data(n,1) > 60 & Data(n,1) <= 75
        variable = Data(n,1)*((111.618-111.412)/15)+111.412;
        LatDistConv = [variable*0.62 (variable*0.62)/60
((variable*0.62)/60)/60];
        y(n,1) = y1(n,:)*LatDistConv';
        n = n+1;
    elseif Data(n,1) > 75 & Data(n,1) <= 90
        variable = Data(n,1)*((111.694-111.618)/15)+111.618;
        LatDistConv = [variable*0.62 (variable*0.62)/60
((variable*0.62)/60)/60];
        y(n,1) = y1(n,:)*LatDistConv';
        n = n+1;
    end
end

% Initialize looping variables and again loop through each
element of the Data matrix to determine the proper conversion
matrix for longitude, LonDistConv. The number of nautical miles
in a degree of longitude decreases as distance from the equator
increases. In order to determine the LonDistConv matrix, first
determine the degree of latitude of the contact, and then
interpolate to determine the conversion matrix.
n = 1;
while n <= length(Data)
    if Data(n,1) <= 15
        variable = Data(n,1)*((107.551-111.320)/15)+111.320;
        LonDistConv = [variable*0.62 (variable*0.62)/60
((variable*0.62)/60)/60];
        x(n,1) = x1(n,:)*LonDistConv';
        n = n+1;
    elseif Data(n,1) > 15 & Data(n,1) <= 30
        variable = Data(n,1)*((96.486-107.551)/15)+107.551;
        LonDistConv = [variable*0.62 (variable*0.62)/60
((variable*0.62)/60)/60];
        x(n,1) = x1(n,:)*LonDistConv';
        n = n+1;
    elseif Data(n,1) > 30 & Data(n,1) <= 45
        variable = Data(n,1)*((78.847-96.486)/15)+96.486;
        LonDistConv = [variable*0.62 (variable*0.62)/60
((variable*0.62)/60)/60];
        x(n,1) = x1(n,:)*LonDistConv';
        n = n+1;
    elseif Data(n,1) > 45 & Data(n,1) <= 60
```

```
        variable = Data(n,1)*((55.800-78.847)/15)+78.847;
        LonDistConv = [variable*0.62 (variable*0.62)/60
((variable*0.62)/60)/60];
        x(n,1) = x1(n,:)*LonDistConv';
        n = n+1;
    elseif Data(n,1) > 60 & Data(n,1) <= 75
        variable = Data(n,1)*((28.902-55.800)/15)+55.800;
        LonDistConv = [variable*0.62 (variable*0.62)/60
((variable*0.62)/60)/60];
        x(n,1) = x1(n,:)*LonDistConv';
        n = n+1;
    elseif Data(n,1) > 75 & Data(n,1) <= 90
        variable = Data(n,1)*((0-28.902)/15)+28.902;
        LonDistConv = [variable*0.62 (variable*0.62)/60
((variable*0.62)/60)/60];
        x(n,1) = x1(n,:)*LonDistConv';
        n = n+1;
    end
end

% Initialize looping variables.
cse = c;
n = 1;
loop = length(c);

% Loop through the matrix to convert course headings in degrees
into units of radians on the unit circle.
while n <= loop
    if c(n) == 0
        c(n) = c(n)*(pi/180)+(pi/2);
    elseif c(n) == 90
        c(n) = c(n)*(pi/180)-(pi/2);
    elseif c(n) == 180
        c(n) = c(n)*(pi/180)+(pi/2);
    elseif c(n) == 270
        c(n) = c(n)*(pi/180)-(pi/2);
    elseif c(n)>0 & c(n)<90
        c(n) = (pi/2)-(c(n)*(pi/180));
    elseif c(n)>90 & c(n)<180
        c(n) = (pi/2)-(c(n)*(pi/180));
    elseif c(n)>180 & c(n)<270
        c(n) = (450-c(n))*(pi/180);
    elseif c(n)>270 & c(n)<360
        c(n) = (450-c(n))*(pi/180);
    end
n = n + 1;
end

% Determine dx and dy components of velocity as a function of
time
dx(:,1) = (s/10).*cos(c);
dy(:,1) = (s/10).*sin(c);

% Call the TimeFuseReal function to convert the data timestamps
into time-steps.
timestep = TimeFuseReal(Data(:,9),default);
```

```
% Store the finalized data in the A matrix to be sent back to the
RealWorld function for analysis.
A = [x y dx dy cse s counter timestep];
```

```
function [compold,Data,maxtime,clusters1] =
ProxKin(A,default,climit,...
    spdlimit,dlimit,spdinput,storage)

% ProxKin    Proximity and kinematic filtering function.
%   ProxKin uses k-means clustering to proximity filter the data
and then kinematically filter those results by determining
similarity of vessel courses, speeds, and the distance between
them.
% Created by LT Kristofer Tester, USN, April 2013

% Determine the maximum time-step that has been assigned to the
data
maxtime = max(A(:,8));

% Initialize looping variables
k = [];
compar = [];
compold = [];
time = 1;
Data = [];
clusters1 = [];

while time <= maxtime
    clusters = [];
    B = [];
    sizeA = size(A);
    n = 1;
    while n <= sizeA(1)
        if A(n,8) == time;
            B = [B;A(n,:)];
        else end
        n = n+1;
    end
    sizeB = size(B);
    if sizeB(1) >= 2

        % Ensure no global identifier is reported more than one
           time during each time-step. If one is, use the
           deconflict function to resolve the issue.
        B = deconflict(B,time);
        Data = [Data;B];
        [clusters] =
KDVC(B,clusters,time,climit,dlimit,spdlimit);
        sizeclusters = size(clusters);
        clusters1 = [clusters1;clusters];
        if any(clusters)
            sizeclusters = size(clusters);
            maxcluster = max(clusters(:,8));
            j = 1;
            while j <= maxcluster
                h = 1;
                while h <= sizeclusters(1)
                    if clusters(h,8) == j
                        k = [k;clusters(h,7)];
                    else
                    end
                    h = h+1;
                end
                if time == 1
                    if j == 1
                        compar = [compar,k];
                    else
                        sizek = size(k);
                        sizecompar = size(compar);
                        if sizek(1) == sizecompar(1)
                            compar = [compar,k];
                        elseif sizek(1) > sizecompar(1)
                            m = sizek(1)-sizecompar(1);
                            n = 1;
                            while n <= m
                                compar =
    [compar;zeros(1,sizecompar(2))];
                                n = n+1;
                            end
                            compar = [compar,k];
                        else sizek(1) < sizecompar(1)
                            m = sizecompar(1)-sizek(1);
                            n = 1;
                            while n <= m
                                k = [k;0];
                                n = n+1;
                            end
                            compar = [compar,k];
                        end
                    end
                else time ~= 1
                    if j == 1
                        compar = [compar,k];
                    else
                        sizek = size(k);
                        sizecompar = size(compar);
                        sizecompold = size(compold);
                        if sizek(1) == sizecompar(1)
                            compar = [compar,k];
                        elseif sizek(1) > sizecompar(1)
                            m = sizek(1)-sizecompar(1);
                            n = 1;
                            while n <= m
                                compar =
    [compar;zeros(1,sizecompar(2))];
                                n = n+1;
                            end
                            compar = [compar,k];
                        else sizek(1) < sizecompar(1)
                            m = sizecompar(1)-sizek(1);
                            n = 1;
                            while n <= m
                                k = [k;0];
                                n = n+1;
                            end
                            compar = [compar,k];
                        end
                        sizecompar = size(compar);
                    end
                end
                k = [];
                j = j+1;
            end
            compold{time} = compar;
            compar = [];
            clc
        else
        end
    else
    end
    time=time+1;
end
```

```
function B = deconflict(B,time)

% DECONFLICT    Deconflict multiple contact reports.
%   B = DECONFLICT(B,time) returns the B matrix ensuring that no
contact global identifier is reported more than one time per
time-step, which is denoted in the variable time.
%   Created by LT Kristofer Tester, USN, April 2013

% Note - The B matrix is the A matrix, just named differently

% Determine if any global identi0fiers are repeated in the B
matrix, and if so, which ones
uniqueB = unique(B(:,7));
if length(uniqueB) > 1
    countOfB = hist(B(:,7),uniqueB);
    indexToRepeatedValue = (countOfB~=1);
    repeatedValues = uniqueB(indexToRepeatedValue);
else
    repeatedValues = uniqueB;
end

% Determine the size of the B matrix
sizeB = size(B);

% Initialize looping variables
current = 0;
m = 1;
spot = [];
boom = 0;

% Loop through the repeated global identifiers and either:
%     1. If the distance between reports is less than 0.5
nautical miles, average the contacts as one.
%     2. If the distance between reports is greater than 0.5
nautical miles, keep the contacts separate and give one of them a
new global identifier.
while m <= length(repeatedValues)
    C = [];
    spot = [];
    n = 1;
    while n <= sizeB(1)
        if B(n,7) == repeatedValues(m);
            C = [C;B(n,:)];
            spot = [spot;n];
        else end
        n = n+1;
    end
    sizeC = size(C);
    if sizeC(1) == 2
        if sqrt((C(1,1)-C(2,1))^2+(C(1,2)-C(2,2))^2) <= 20;

            % Average the two contacts' components
            avgxpos = (C(1,1)+C(2,1))/2;
            avgypos = (C(1,2)+C(2,2))/2;
            avgdx = (C(1,3)+C(2,3))/2;
            avgdy = (C(1,4)+C(2,4))/2;
            avgcse = (C(1,5)+C(2,5))/2;
            avgspd = (C(1,6)+C(2,6))/2;
            avglon = (C(1,9)+C(2,9))/2;
            avglat = (C(1,10)+C(2,10))/2;
            if avgxpos == 0
                avgxpos = 0.0001;
            else
            end
            if avgypos == 0
                avgypos = 0.0001;
            else
            end
            if avgdx == 0
                avgdx = 0.0001;
            else
            end
            if avgdy == 0
                avgdy = 0.0001;
            else
            end
            if avgcse == 0
                avgcse = 0.0001;
            else
            end
            if avgspd == 0
                avgspd = 0.0001;
            else
            end
            if avglon == 0
                avglon = 0.0001;
            else
            end
            if avglat == 0
                avglat = 0.0001;
            else
            end
            B(spot(1),:) = [avgxpos avgypos avgdx avgdy avgcse
avgspd...
                repeatedValues(m) time avglon avglat];
            B(spot(2),:) = B(spot(2),:)*0;
            sizeB = size(B);
        else
        end
        sizeB = size(B);
    else
        sizespot = size(spot);
        a = 1;
        while a <= sizespot(1)
            if any(spot(a));
                b = a+1;
                while b <= sizespot(1)
                    if C(a,7) ~= 0 & C(b,7) ~= 0
                        dist = sqrt((C(a,1)-C(b,1))^2+(C(a,2)-
C(b,2))^2);
                        if dist <= 20;

                            % Average the contacts' components
                            avgxpos = (C(a,1)+C(b,1))/2;
                            avgypos = (C(a,2)+C(b,2))/2;
                            avgdx = (C(a,3)+C(b,3))/2;
                            avgdy = (C(a,4)+C(b,4))/2;
                            avgcse = (C(a,5)+C(b,5))/2;
                            avgspd = (C(a,6)+C(b,6))/2;
                            avglon = (C(1,9)+C(2,9))/2;
                            avglat = (C(1,10)+C(2,10))/2;
                            if avgxpos == 0
                                avgxpos = 0.0001;
                            else
                            end
                            if avgypos == 0
                                avgypos = 0.0001;
                            else
                            end
                            if avgdx == 0
                                avgdx = 0.0001;
                            else
                            end
                            if avgdy == 0
                                avgdy = 0.0001;
                            else
                            end
                            if avgcse == 0
                                avgcse = 0.0001;
                            else
                            end
                            if avgspd == 0
                                avgspd = 0.0001;
                            else
                            end
                            if avglon == 0
                                avglon = 0.0001;
                            else
                            end
                            if avglat == 0
                                avglat = 0.0001;
                            else
                            end
                            B(spot(a),:) = [avgxpos avgypos avgdx
avgdy...
                                avgcse avgspd repeatedValues(m)
time...
                                avglon avglat];
                            B(spot(b),:) = B(spot(b),:)*0;
                            C(a,:) = B(spot(a),:);
                            current = 1;
                            spot(b) = 0;
                            if length(nonzeros(spot)) <= 1
                                a = a+sizeC(1);
                                b = b+sizeC(1);
                            else
                            end
                            sizeB = size(B);
                            b = b+1;
                        else
                            b = b+1;
                        end
                    else
                        b = b+1;
                    end
                end
                if current == 1
                    spot(a) = 0;
                    current = 0;
                else
                    current = 0;
                end
                a = a+1;
            else
                a = a+1;
            end
        end

        % Create a new contact global identifier
        sizespot = size(spot);
        j = 1;
        sizeB = size(B);
    end
    m = m+1;
end

% Redefine the B matrix for output. This B matrix will have no
repeated global identifier values.
B = [nonzeros(B(:,1)) nonzeros(B(:,2)) nonzeros(B(:,3))...
    nonzeros(B(:,4)) nonzeros(B(:,5)) nonzeros(B(:,6))...
    nonzeros(B(:,7)) nonzeros(B(:,8)) nonzeros(B(:,9))...
    nonzeros(B(:,10))];
```

```
function [clusters] =
KDVC(A,clusters,time,climit,dlimit,spdlimit)

% KDVC   K-means, distance, and displacement clustering.
%   [clusters] = KDVC(A,clusters,time,climit,dlimit,spdlimit)
returns the kinematic clusters at a given time-step based on user
inputs where A is the input dataset, clusters is the matrix for
output storage, time is the time-step being evaluated, climit is
the user defined course limit in degrees, dlimit is the user
defined distance limit in nautical miles, and spdlimit is the
user definied speed limit in knots.
%   Created by LT Kristofer Tester, USN, April 2013

% Determine the size of the input dataset, A
sizeA = size(A);

% Determine the sample of contacts from A to be used in
determining the centroids of k clusters using subset-farthest-
first methodology
sample = round(sizeA(1));
syms w
k = round(solve(sample == 2*w*log(w)));

% Call for function sff in which subset-farthest-first
methodology is implemented. Returns a matrix, kstart, which
contains the centroids for k-means clustering.
if sample >= 3
    eval(['[kstart] = CentroidSff(A,k,sample)']);
else
    kstart = A(1:sample,:);
end

% Conduct k-means clustering on the dataset A using the centroids
stored in kstart. Store the results as B
sizekstart = size(kstart);
if sizekstart(1) == sizeA(1)
    if kstart ~= A
        B = kmeans(A,[],'start',kstart,'EmptyAction','drop');
    else
        B = ones(sizekstart(1),1);
    end
else
            B = kmeans(A,[],'start',kstart,'EmptyAction','drop');
end

% Redefine the A matrix to incorporate the results of k-means
clustering, B
A = [A(:,1) A(:,2) A(:,3) A(:,4) A(:,5) A(:,6) A(:,7) B A(:,8)
A(:,9) A(:,10)];

% Initialize loop variable which aids in titling of the plots
provided by the function
h = 1;

% Define the default line style for plotting
set(0,'DefaultAxesLineStyleOrder',{'-','--',':'})
```

```
% Begin loop for clustering analysis of A matrix and continue
loop as long as A has non-zero members
leginfo = {};
while any(A) == 1

    % Determine the row of the A matrix that has the maximum
speed value.
    % Use the corresponding contact as the seed contact.
    [a,b] = max(A(:,6));
    c = A(b,:);
    A(b,:) = A(b,:)*0;

    % Call function Kinematic which is the actual clustering
algorithm. Returns the A matrix along with those contacts that
are clustered together in c
    eval(['[A,c] = Kinematic(A,c,climit,dlimit,spdlimit,h)']);

    % Determine the size of the c matrix
    sizec = size(c);

    % If contacts were clustered together, drop in and plot those
contacts. If contacts were not clustered together, plot nothing
and loop to
    if sizec(1) > 1

        % Ensure the contacts in c are sorted by global
identifier in ascending order
        update = c;
        update = sortrows(update,7);
        clusters = [clusters;update];

        % Quiver plot the contacts that are clustered together in
each time-step
        quiver(c(:,10),c(:,11),c(:,3),c(:,4));
        title('Kinematic Microclusters')
        xlabel('Position in miles from the origin')
        ylabel('Position in miles from the origin')
        leginfo{h} = ['c_' num2str(h)];
        grid on
        hold on
        set(gcf,'color','w');
        h = h + 1;

        % Determine the size of the matrix clusters and add a row
of zeros to separate each cluster found in each time-step
        sizeclusters = size(clusters);
        clusters = [clusters;zeros(1,sizeclusters(2))];
    else sizec(1) < 1
    end
end

% Provide the legend information for the cluster plot
if isempty(leginfo) == 0
    legend(leginfo)
else
end
hold off
```

```
clc
```

```
function [kstart] = CentroidSff(A,k,sample)

% SFF    Subset farthest first function
%   [kstart] = SFF(A,k,sample) returns the centroid positions for
use in k-means clustering that are determined using subset
farthest first technology, where A is the input data matrix, k is
the number of centroids to determine, and sample is the size of
the subset of the A matrix to be considered.
%   Created by LT Kristofer Tester, USN, April 2013

% Initialize looping variables
n = 1;
j = 1;
sizeA = size(A);
z = sizeA(1);
l = round(sizeA(1)/2-k/2);
B = A;

% Initialize a looping variable and assign the first centroid to
kstart
i = 1;
kstart(1,:) = B(1,:);
B(1,:) = B(1,:)*0;

% After assigning the first member of kstart, reconfigure the B
matrix to exhaust that member
x = nonzeros(B(:,1));
y = nonzeros(B(:,2));
dx = nonzeros(B(:,3));
dy = nonzeros(B(:,4));
cse = nonzeros(B(:,5));
spd = nonzeros(B(:,6));
counter = nonzeros(B(:,7));
t = nonzeros(B(:,8));
lon = nonzeros(B(:,9));
lat = nonzeros(B(:,10));
B = [x y dx dy cse spd counter t lon lat];

% Loop through the B matrix and assign follow-on members of the
kstart matrix (centroids) as the contact that is farthest from
the previous selection.
while i < k
    n = 1;
    sizeB = size(B);
    while n < sizeB(1)
        d(n,i) = sqrt((B(n,1)-kstart(i,1)).^2 + (B(n,2)-
kstart(i,2)).^2);
        n = n+1;
    end
    [F,I] = max(d(:,i));
    i = i+1;
    kstart(i,:) = B(I,:);
    B(I,:) = B(I,:)*0;

    % Reconfigure the B matrix to account for kstart assignment
and exhaust further possible consideration of the members.
    x = nonzeros(B(:,1));
    y = nonzeros(B(:,2));
    dx = nonzeros(B(:,3));
    dy = nonzeros(B(:,4));
    cse = nonzeros(B(:,5));
    spd = nonzeros(B(:,6));
    counter = nonzeros(B(:,7));
    t = nonzeros(B(:,8));
    lon = nonzeros(B(:,9));
    lat = nonzeros(B(:,10));
    B = [x y dx dy cse spd counter t lon lat];
end
```

```
function [A,c] = Kinematic(A,c,climit,dlimit,spdlimit,h)

% Kinematic    Distance and displacement clustering.
%   [A,c] = Kinematic(A,c,climit,dlimit,spdlimit,h) returns the
kinematic clusters at a given time-step based on user inputs
where A is the input dataset, c is the seed contact, climit is
the user defined course limit in degrees, dlimit is the user
defined distance limit in nautical miles, spdlimit is the user
definied speed limit in knots, and h is the time-step.
%   Created by LT Kristofer Tester, USN, April 2013

% Determine the size of the A matrix and initialize looping
variable
sizeA = size(A);
n = 1;

% Compare each row of the A matrix to the seed contact in c and
determine if:
%     1. They have the same k-means cluster assignment
%     2. The distance between them is less than the user defined
threshold
%     3. The difference in their courses is less than the user
defined threshold
%     4. The difference in their speeds is less than the user
defined threshold
% If so, cluster them together, set the row in the A matrix to
zero to exhaust it from further consideration, and store the
contact in c
while n <= sizeA(1)
    if c(1,8) == A(n,8);
        if sqrt((c(1,1)-A(n,1)).^2 + (c(1,2)-A(n,2)).^2) <
dlimit;
            if (A(n,5) >= c(1,5)-climit) & (A(n,5) <=
c(1,5)+climit);
                if (A(n,6) >= c(1,6)-spdlimit) & (A(n,6) <=
c(1,6)+...
                        spdlimit);
                    c(n+1,:) = A(n,:);
                    A(n,:) = A(n,:)*0;
                else end
            else end
        else end
    else end
    n = n+1;
end
x = nonzeros(c(:,1));
y = nonzeros(c(:,2));
dx = nonzeros(c(:,3));
dy = nonzeros(c(:,4));
cse = nonzeros(c(:,5));
spd = nonzeros(c(:,6));
counter = nonzeros(c(:,7));
sizex = size(x);
h = h*ones(sizex(1),1);
t = nonzeros(c(:,9));
Lon = nonzeros(c(:,10));
Lat = nonzeros(c(:,11));
```

```
% Redefine the c matrix to account for those contacts deemed to
be clustered with the seed contact
c = [x y dx dy cse spd counter h t Lon Lat];
end
```

```
function
[MovingClusters,FuseClusters,DiffClusters,CurrentClusters,...
    timecount] = Temporal(compold,storage)

% Temporal    Temporal analysis function of STC algorithm.
%   Temporal compares the kinematic cluster snapshots from each
time-step to determine which clusters are moving through time.
The results are stored in MovingClusters. FuseClusters contains
the continuity of the moving clusters. DiffClusters contains the
vessel identifiers of those vessels that join or depart already
formed moving clusters. CurrentClusters contains the kinematic
clusters found at the last time-step of data. Created by LT
Kristofer Tester, USN, April 2013

% A snapshot of each time-step has been analyzed, and the
kinematic clusters that occur at each time-step are stored in the
cell array compold. Now initialize looping variables and loop
through compold to determine if any kinematic clusters were
stored. If there were, assign a value of one to the variable
moveon.
zz = 1;
moveon = 0;
timecount = 0;
timemat = [];
while zz <= length(compold)
    if any(compold{zz})
        moveon = 1;
        timecount = timecount+1;
        timemat = [timemat;zz];
        zz = zz+1;
    else
        zz = zz+1;
    end
end

% With moveon equal to one it is known that there are kinematic
clusters stored in compold. It is now time to perform temporal
analysis on those kinematic clusters and determine if any of them
move together over time. Again, while looping through the time-
steps that contain kinematic clusters, the variable fuse will be
defined as the intersection of the current time-step's kinematic
clusters' memberships with the next time-step's kinematic
clusters' memberships. If fuse is greater than or equal to a user
defined threshold, store the kinematic cluster as a moving
cluster. If fuse is less than the threshold or zero, compare the
current time-step's kinematic clusters' membership to the next
time-step plus one's clusters' membership in order to recalculate
fuse. If fuse is still less than the threshold or zero, discard
the kinematic cluster from consideration of being a moving
cluster.
if moveon == 1
    timemat = sort(timemat);
    mm = 0;
    sizetimemat = size(timemat);
    time = timemat(1);
    MovingClusters = {};
    CurrentClusters = {};
    FuseClusters = {};
    DiffClusters = {};
    while time <= timemat(sizetimemat(1))
        mm = mm+1;
        if time == timemat(1);
            B = compold{time};
            C = compold{timemat(2)};
        else end
        if time <= timemat(sizetimemat(1)-2);
            D = compold{timemat(mm+2)};
        else end
        columnB = size(B);
        columnC = size(C);
        columnD = size(D);
        j = 1;
        while j <= columnB(2)
            h = 1;
            sizeB = size(nonzeros(B(:,j)));
            if time <= timemat(sizetimemat(1)-1);
                while h <= columnC(2)
                    sizeC = size(nonzeros(C(:,h)));
                    Z =
intersect(nonzeros(B(:,j)),nonzeros(C(:,h)));
                    sizeZ = size(Z);
                    if sizeC(1) > sizeB(1);
                        fuse = sizeZ(1)/sizeC(1);
                    elseif sizeB(1) > sizeC(1);
                        fuse = sizeZ(1)/sizeB(1);
                    else sizeC(1) == sizeB(1);
                        fuse = sizeZ(1)/sizeB(1);
                    end
                    katysnumber = 0;
                    if fuse >= storage;
                        MovingClusters{time,j} = B(:,j);

                        FuseClusters{time,j} = fuse;
                        if fuse ~= 1 & fuse ~= 0
                            X = setdiff(B(:,j),C(:,h));
                            DiffClusters{time,j} = X;
                        else
                        end
                        h = columnC(2)+1;
                        katysnumber = 1;
                    else
                        if time <= timemat(sizetimemat(1)-2) & h
<=...
                                columnD(2);
                            sizeD = size(nonzeros(D(:,h)));
                            Z = intersect(nonzeros(B(:,j)),...
                                nonzeros(D(:,h)));
                            sizeZ = size(Z);
                            if sizeD(1) > sizeB(1);
                                fuse = sizeZ(1)/sizeD(1);
                            elseif sizeB(1) > sizeD(1);
                                fuse = sizeZ(1)/sizeB(1);
                            else sizeD(1) == sizeB(1);
                                fuse = sizeZ(1)/sizeB(1);
                            end
                            if fuse >= storage;
                                MovingClusters{time,j} = B(:,j);

                                FuseClusters{time,j} = fuse;
                                if fuse ~= 1 & fuse ~= 0
                                    X = setdiff(B(:,j),D(:,h));
                                    DiffClusters{time,j} = X;
                                else
                                end
                                h = columnD(2)+1;
                                katysnumber = 1;
                            else end
                        else end
                    end
                    h = h+1;
                end
            else time == timemat(sizetimemat(1));
                CurrentClusters{1,j} = B(:,j);
            end
            j = j+1;
        end
        B = C;
        C = D;
        if mm < sizetimemat(1)
            time = timemat(mm+1);
        else
            time = max(timemat)+1;
        end
    end
else
    return
end
```

```
function
Postprocessing(MovingClusters,FuseClusters,DiffClusters,...
    CurrentClusters,timecount,Data,maxtime,clusters1)

% Postprocessing    Function to handle the development of usable
text and visual outputs.
%   Postprocessing creates user-friendly text and visual outputs
from the cell array inputs to provide the user an understanding
of the algorithm’s findings.
% Created by LT Kristofer Tester, USN, April 2013

clc
moveon = 0;
zz = 1;
sizeMovingClusters = size(MovingClusters);
while zz <= sizeMovingClusters(1)*sizeMovingClusters(2)
    if any(MovingClusters{zz})
        moveon = 1;
        zz = sizeMovingClusters(1)*sizeMovingClusters(2)+1;
    else
        zz = zz+1;
    end
end
if moveon == 1
    CompClusters = MovingClusters;
    occur = [];
    begin = {};
    time = 1;
    index = 1;
    B = {};
    cluster1 = [];
    l = 0;
    str = ('Moving Clusters:');
    disp(str)
    str = (' ');
    disp(str)
    while time <= sizeMovingClusters(1)
        h = 1;
        while h <= sizeMovingClusters(2)
            if any(MovingClusters{time,h})
                b = nonzeros(MovingClusters{time,h});
                index = 1;
                occur = [];
                while index <=
sizeMovingClusters(1)*sizeMovingClusters(2)
                    if any(intersect(MovingClusters{index},b))
                        J = nonzeros(MovingClusters{index});
                        sizeindex = size(J);
                        sizeb = size(b);
                        if sizeindex(1) == sizeb(1)
                            if J == b
                                [I,J] =
ind2sub(sizeMovingClusters,index);
                                occur = [occur;I];
                                MovingClusters{index} =...
                                    MovingClusters{index}*0;
                                index = index+1;
                            else
                                index = index+1;
                            end
                        else
                            index = index+1;
                        end
                    else
                        index = index+1;
                    end
                end
                if any(occur)
                    l = l+1;
                    occur = sort(occur);
                    sizeoccur = size(occur);
                    b = nonzeros(b);
                    sizeb = size(b);
                    sizeCurrentClusters = size(CurrentClusters);
                    current = 0;
                    q = 1;
                    while q <= sizeCurrentClusters(2)
                        Q = nonzeros(CurrentClusters{q});
                        sizeq = size(Q);
                        if sizeq(1) == sizeb(1)
                            if nonzeros(CurrentClusters{q}) ==
nonzeros(b)
                                current = 1;
                                q = sizeCurrentClusters(2)+1;
                            else
                                q = q+1;
                            end
                        else
                            q = q+1;
                        end
                    end
                    if current == 1;
                        occur = [occur;maxtime];
                        if length(occur) == max(occur)-
min(occur)+1;
                            str = ['Cluster ' num2str(l)...
                                ' containing contacts
',num2str(b'),...
                                ' begins at time
',num2str(occur(1)),...
                                ' and is a current cluster'];
                        else
                            str = ['Cluster ' num2str(l)...
                                ' containing contacts
',num2str(b'),...
                                ' begins at time
',num2str(occur(1)),...
                    ', is a current cluster, but gains or loses
members'];
                        end
                        disp(str)
                        current = 0;
                    else
                        str = ['Cluster ' num2str(l)...
```

```
                                        ' containing contacts
',num2str(b'),...
                                        ' begins at time ',
num2str(occur(1)),...
                            ' and ends at time ',
num2str(occur(sizeoccur(1)))];
                                disp(str)
                            end

                            % Average the cluster-contacts into a
                              representative cluster at each timestep
                              they occur in order to store the
                              information for presentation on the visual
                              output.
                            sizeoccur = size(occur);
                            mm = 0;
                            y = occur(1);
                            while y <= max(occur)
                                mm = mm+1;
                                z = 1;
                                clust = [];
                                while z <= sizeb(1)
                                    K = find(Data(:,7)==b(z));
                                    sizeK = size(K);
                                    if sizeK == 1;
                                        clust = [clust;Data(K,:)];
                                        z = z+1;
                                    else
                                        clust = [clust;Data(K(mm),:)];
                                        z = z+1;
                                    end
                                end
                                avgxpos = mean(clust(:,1));
                                avgypos = mean(clust(:,2));
                                avgdx = mean(clust(:,3));
                                avgdy = mean(clust(:,4));
                                avglon = mean(clust(:,9));
                                avglat = mean(clust(:,10));
                                avgcse = mean(clust(:,5));
                                avgspd = mean(clust(:,6));
                                cluster1 = [cluster1;avgxpos avgypos
avgdx avgdy...
                                    y FuseClusters{time,h} l avglon
avglat...
                                    avgcse avgspd];
                                if mm < sizeoccur(1)
                                    y = occur(mm+1);
                                else
                                    y = max(occur)+1;
                                end
                            end
                            d = 1;
                            while d <= length(CurrentClusters)
                                kk = nonzeros(CurrentClusters{d});
                                ll = nonzeros(clust(:,7));
                                sizekk = size(kk);
                                sizell = size(ll);
```

```
                        if sizekk == sizell
                            if length(intersect(nonzeros...

(CurrentClusters{d}),nonzeros...

(clust(:,7))))/length(nonzeros...
                                    (clust(:,7))) == 1
                                f = nonzeros(clust(:,7));
                                g = 1;
                                while g <= length(f)
                                    row = find(clusters1(:,7) ==
f(g),1);
                                    clusters1(row,:) =
clusters1(row,:)*0;
                                    g = g+1;
                                end
                            else end
                            d = d+1;
                        else
                            d = d+1;
                        end
                    end
                    sizecluster1 = size(cluster1);
                    cluster1 =
[cluster1;zeros(1,sizecluster1(2))];
                end
            else
                h = h+1;
            end
        end
        time = time+1;
    end
    sizecluster1 = size(cluster1);
    o = max(clusters1(:,8));
    r = 1;
    b = [];
    l = l+1;

    % Reorganize the matrix cluster1 for administrative purposes.
    % Determine the maximum and minimum of the cardinal
directions for plotting purposes. Call the moneyscatter function
to create an interactive visual output the function
plot_google_map will lay the google map representation of the
area underneath the output plot for better user situational
awareness.
    cluster1 = [cluster1(:,8) cluster1(:,9) cluster1(:,3)
cluster1(:,4)...
        cluster1(:,5) cluster1(:,6) cluster1(:,7)
cluster1(:,10)...
        cluster1(:,11)];
    west = min(nonzeros((cluster1(:,1))));
    east = max(nonzeros((cluster1(:,1))));
    south = min(nonzeros((cluster1(:,2))));
    north = max(nonzeros((cluster1(:,2))));
    PlotScatter(cluster1)
    title('Interactive Visual Cluster Representation')
    xlabel('Degrees of Longitude')
    ylabel('Degrees of Latitude')
    axis([west-0.5 east+0.5 south-0.5 north+0.5]);
    set(gcf,'color','w');
    plot_google_map

    % Using the cell array DiffClusters, determine which contacts
join and depart other moving clusters throughout the timeline of
analysis. This information is then output in text format for the
user to better determine potential contacts of interest.
    str = (' ');
    disp(str)
    disp(str)
    str = ('Contacts of Interest:');
    disp(str)
    str = (' ');
    disp(str)
    h = 1;
    sizeDiffClusters = size(DiffClusters);
    sizeDC = sizeDiffClusters(1)*sizeDiffClusters(2);
    while h <= sizeDC
        if any(DiffClusters{h});
            occur = [];
            m = nonzeros(DiffClusters{h});
            j = 1;
            sizeCompClusters = size(CompClusters);
            sizeCC = sizeCompClusters(1)*sizeCompClusters(2);
            while j <= sizeCC
                if any(intersect(m,CompClusters{j}));
                    [I,J] = ind2sub(sizeCompClusters,j);
                    occur = [occur;I];
                    j = j+1;
                else
                    j = j+1;
                end
            end
            dep = max(occur);% + 1;
            current = 0;
            if length(occur) == timecount-1
                if length(m) > 1
                    str = ['Contacts ' num2str(m')...
    ' are a moving cluster that join and depart other larger
clusters.'];
                    disp(str)
                else
                    str = ['Contact ' num2str(m')...
    ' is a moving cluster that joins and departs other larger
clusters.'];
                    disp(str)
                end
            else
                kk = 1;
                sizem = size(m);
                while kk <= sizem(1)
                    if any(find((Data(:,7) == m(kk)) &
(Data(:,8)...
                                == dep)));
                        s(kk) = find((Data(:,7) == m(kk)) &
(Data(:,8)...
                            == dep));
                        kk = kk+1;
                        current = 1;
                    else
                        kk = kk+1;
                    end
                end
            end
            if current == 1
                course = 0;
                speed = 0;
                sizes = size(s);
                kk = kk-1;
                if sizes(1) == 1;
                    course = Data(s(1),5);
                    speed = Data(s(1),6);
                else
                    while kk >= 1
                        course = (course+Data(s(kk),5))/2;
                        speed = (speed+Data(s(kk),6))/2;
                        kk = kk-1;
                    end
                end
                if length(occur) == max(occur)-min(occur)+1
                    str = ['Contact ' num2str(m')...
                        ' joins a cluster at time '
num2str(min(occur))...
               ' and remains with the cluster until it departs at
time '...
         num2str(max(occur)) '. ''The contact departs on average
course '...
         num2str(round(course)) ' at average speed of '...
         num2str(round(speed)) ' knots.'];
                    disp(str)
                else
                    str = ['Contact ' num2str(m')...
                        ' joins a cluster at time '
num2str(min(occur))...
 ', departs the cluster, and then rejoins it, finally departing
at time '...
 num2str(max(occur)) '. ''The contact departs on average course
'...
 num2str(round(course)) ' at average speed of '
num2str(round(speed))...
 ' knots.'];
                    disp(str)
                end
            else
            end
            h = h+1;
        else
            h = h+1;

        end
    end
else
```

```
      return
   end
```

```
function PlotScatter(cluster1)

% PlotScatter    Interactive scatterplot output
%   PlotScatter(cluster1) returns an interactive scatterplot of
the information stored in the cluster1 matrix. The user is able
to interactively click on the scatterplot to determine
information from various data points.
%   Created by LT Kristofer Tester, USN, April 2013

% Set default line style for the scatterplot
set(0,'DefaultAxesLineStyleOrder',{'-','-','-','-'})

% Open new figure
fh = figure();

% Plot various colors on a far reach of the figure to enable
labeling in the legend
h1 = plot(1e6,1e6,'r');
hold on
h2 = plot(1e6,1e6,'color',[1 .5 0]);
h3 = plot(1e6,1e6,'g');
h4 = plot(1e6,1e6,'color',[0 .5 .5]);
h5 = plot(1e6,1e6,'color',[.25 0 1]);

% Intialize looping variables
n = 1;
xy = [];

% Loop through each row of cluster1 and plot each cluster in the
appropriate color and style according to its attributes
sizecluster1 = size(cluster1);
while n < sizecluster1(1)
    if any(cluster1(n,:))
        if cluster1(n+1,5) == 0;
            if n == 1 || (cluster1(n+1,5) == 0 & cluster1(n-1,5)
== 0)
                if cluster1(n,6) == 1;
                    if any(xy)
                        xy = [xy;cluster1(n,1) cluster1(n,2)];
                        plot(xy(:,1),xy(:,2),'rx-');
                        hold on
                        grid on
                        xy = [];
                    else end
                    plot(cluster1(n,1),cluster1(n,2),'ro')
                    hold on
                    grid on
                elseif cluster1(n,6) >= 0.75 & cluster1(n,6) < 1;
                    if any(xy)
                        xy = [xy;cluster1(n,1) cluster1(n,2)];
                        plot(xy(:,1),xy(:,2),'x-','color',[1 .5
0]);
                        hold on
                        grid on
                        xy = [];
                    else end
```

```
plot(cluster1(n,1),cluster1(n,2),'o','color',[1 .5 0])
                    hold on
                    grid on
                elseif cluster1(n,6) >= 0.5 & cluster1(n,6) <
0.75;
                    if any(xy)
                        xy = [xy;cluster1(n,1) cluster1(n,2)];
                        plot(xy(:,1),xy(:,2),'gx-');
                        hold on
                        grid on
                        xy = [];
                    else end
                    plot(cluster1(n,1),cluster1(n,2),'go')
                    hold on
                    grid on
                elseif cluster1(n,6) >= 0.25 & cluster1(n,6) <
0.5;
                    if any(xy)
                        xy = [xy;cluster1(n,1) cluster1(n,2)];
                        plot(xy(:,1),xy(:,2),'x-','color',[0 .5
.5]);

                        hold on
                        grid on
                        xy = [];
                    else end

plot(cluster1(n,1),cluster1(n,2),'o','color',[0 .5 .5])
                    hold on
                    grid on
                elseif cluster1(n,6) > 0 & cluster1(n,6) < 0.25;
                    if any(xy)
                        xy = [xy;cluster1(n,1) cluster1(n,2)];
                        plot(xy(:,1),xy(:,2),'x-','color',[.25 0
1]);

                        hold on
                        grid on
                        xy = [];
                    else end

plot(cluster1(n,1),cluster1(n,2),'o','color',[.25 0 1])
                    hold on
                    grid on
                else cluster1(n,6) == 0
                    scatter(cluster1(n,1),cluster1(n,2),'ko')
                    hold on
                    grid on
                end
                n = n+1;
            else
                if cluster1(n-1,6) == 1;
                    if any(xy)
                        xy = [xy;cluster1(n,1) cluster1(n,2)];
                        plot(xy(:,1),xy(:,2),'rx-');
                        hold on
                        grid on
                        xy = [];
                    else end
                    plot(cluster1(n,1),cluster1(n,2),'ro')
                    hold on
                    grid on
                elseif cluster1(n-1,6) >= 0.75 & cluster1(n,6) <
1;
                    if any(xy)
                        xy = [xy;cluster1(n,1) cluster1(n,2)];
                        plot(xy(:,1),xy(:,2),'x-','color',[1 .5
0]);
                        hold on
                        grid on
                        xy = [];
                    else end

plot(cluster1(n,1),cluster1(n,2),'o','color',[1 .5 0])
                    hold on
                    grid on
                elseif cluster1(n-1,6) >= 0.5 & cluster1(n,6) <
0.75;
                    if any(xy)
                        xy = [xy;cluster1(n,1) cluster1(n,2)];
                        plot(xy(:,1),xy(:,2),'gx-');
                        hold on
                        grid on
                        xy = [];
                    else end
                    plot(cluster1(n,1),cluster1(n,2),'go')
                    hold on
                    grid on
                elseif cluster1(n-1,6) >= 0.25 & cluster1(n,6) <
0.5;
                    if any(xy)
                        xy = [xy;cluster1(n,1) cluster1(n,2)];
                        plot(xy(:,1),xy(:,2),'x-','color',[0 .5
.5]);
                        hold on
                        grid on
                        xy = [];
                    else end

plot(cluster1(n,1),cluster1(n,2),'o','color',[0 .5 .5])
                    hold on
                    grid on
                elseif cluster1(n-1,6) > 0 & cluster1(n,6) <
0.25;
                    if any(xy)
                        xy = [xy;cluster1(n,1) cluster1(n,2)];
                        plot(xy(:,1),xy(:,2),'x-','color',[.25 0
1]);
                        hold on
                        grid on
                        xy = [];
                    else end

plot(cluster1(n,1),cluster1(n,2),'o','color',[.25 0 1])
                    hold on
                    grid on
                else cluster1(n-1,6) == 0
                    scatter(cluster1(n,1),cluster1(n,2),'ko')
                    hold on
                    grid on
                end
                n = n+1;
            end
        elseif cluster1(n+1,5) ~= 0;
            xy = [xy;cluster1(n,1) cluster1(n,2)];
            n = n+1;
        end
    else
        n = n+1;
    end
end

% Define the legend properties and labels for each color
legend([h1 h2 h3 h4 h5],'100%','75-99%','50-74%;','25-49%','0-
25%');
% Enable data cursor mode in the figure and call the function to
enable interactive use and response from the figure
dcm = datacursormode(fh);
datacursormode on
set(dcm,'displaystyle','window');
set(dcm,'UpdateFcn',{@poscluster,cluster1})

% Function call to provide the information for data points on the
scatterplot based on where the data cursor is.
function InfoBox = poscluster(obj,event_obj,cluster1)

% Determine the position of the user mouse click and store in pos
pos = get(event_obj,'Position');

% Define the portions of pos that correspond to the x and y
position values
x = pos(1);
y = pos(2);

% Search the cluster1 matrix for the contact in the selected x
and y posiition
a = find(cluster1(:,1) == x);
b = find(cluster1(:,2) == y);
c = find(cluster1(:,5) ~= 0);

% Set the variable row to the contact in the given x and y
position
row = intersect(a,b);
row = intersect(row,c);

% Set output text accordingly (Could be anything we want it to
be)
InfoBox = {['Longitude: ' num2str(pos(1),4)],...
    ['Latitude: ' num2str(pos(2),4)]...
    ['Cluster: ' num2str(cluster1(row,7))]...
    ['Time: ' num2str(cluster1(row,5))]...
    ['Average Course: ' num2str(round(cluster1(row,8)))]...
    ['Average Speed: ' num2str(round(cluster1(row,9)))]};
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] Joint Chiefs of Staff, "Department of Defense Dictionary of Military and Associated Terms," Department of Defense, Washington, DC. [Online]. Available: http://www.dtic.mil/doctrine/dod_dictionary/

[2] Joint Chiefs of Staff, "Command and Control for Joint Maritime Operations," Department of Defense, Washington, DC. [Online]. Available: http://www.dtic.mil/doctrine/new_pubs/jp3_32.pdf

[3] Chief of Naval Operations, "Maritime Domain Awareness," Department of the Navy, Norfolk, VA. [Online]. Available: https://community.apan.org/.../TM-3_2D00_32.1_2D00_10.pdf

[4] Chief of Naval Operations, "A Cooperative Strategy for 21st Century Seapower," Department of the Navy, Washington, DC. [Online]. Available: http://www.navy.mil/maritime/Maritimestrategy.pdf

[5] S. Das, *et al.*, "Spatiotemporal clustering for aggregating hostile units in cluttered environments," in 9th International Conference on Information Fusion, pp.1,8, 10–13, 2006.

[6] P. Kanjilal, *et al.*, "Spatiotemporal clustering from noisy data," presented at the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, Jan. 2007.

[7] S. Das, *High-Level Data Fusion*, Norwood, MA: Artech House, 2008.

[8] G. Welch and G. Bishop, "An introduction to the Kalman filter," UNC, Chapel Hill, NIC. Tech. Rep. TR95-041, Nov. 2000. [Online]. Available: http://www.cs.unc.edu.

[9] J. K. Tunaley, "Algorithms for ship detection and tracking using satellite imagery," in *Geoscience and Remote Sensing Symposium,* IEEE International Proc., vol. 3, Sept. 20-24 2004.

[10] J. Hwang, *et al.*, "Searching for similar trajectories on road networks using spatio-temporal similarity," in *Proc. of the 10th East European Conference on Advances in Databases and Information Systems*, Berlin, Germany: Springer-Verlag, pp. 282–295, 2006.

[11] Y. Si, *et al.*, (2009) "Spatio-temporal dynamics of global outbreaks match bird migration patterns," in *Geospat Health*, vol. 4, pp. 65–78, 2009.

[12] Y. Yuan and M. Raubal, “Spatio-temporal knowledge discovery from georeferenced mobile phone data,” presented at the Proc. Workshop Movement Pattern Analysis, Zurich, Switzerland, Sept. 14, 2010.

[13] D. C. Eckley and K. M. Curtin, “Evaluating the spatiotemporal clustering of traffic incidents,” in *Computers, Environment and Urban Systems*, vol. 37, pp. 70-81, Jan. 2013.

[14] *Memorandum on Navy Maritime Domain Awareness Concept,* Department of the Navy, Washington, DC, Tech. Memo, May 29, 2007.

[15] S. Kisilevich, *et al.*, *Spatio-Temporal Clustering, Data Mining and Knowledge Discovery Handbook*, 2nd edition, New York: Springer Press, 2010.

[16] A. Vadivel, *et al.*, “Performance comparison of distance metrics in content-based image retrieval applications,” presented at the International Conference on Information Technology, Newark, NJ, 2003.

[17] P. P. Teodorescu, “Kinematics," in *Mechanical Systems, Classical Models: Particle Mechanics*. Dordrecht, Netherlands: Springer, ch. 5, pp. 287, 2006.

[18] S. P. Lloyd, “Least squares quantization in PCM,” IEEE Transactions on Information Theory, vol. 28, pp. 129–137, Jan. 1982.

[19] J. B. MacQueen, “Some Methods for classification and Analysis of Multivariate Observations,” Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability vol. 1: Statistics, Berkeley: University of California Press. pp. 281–297, 1963.

[20] D. MacKay, “An Example Inference Task: Clustering,” in *Information Theory, Inference and Learning Algorithms*, Cambridge, United Kingdom: Cambridge University Press, pp. 284–292, Oct. 6, 2003.

[21] MathWorks, “MATLAB help topic on k-means clustering, R2013a Documentation,” MathWorks, Natick, MA. [Online]. Available: http://www.mathworks.com/help/stats/kmeans.html

[22] International Maritime Organization, “International Convention for the Safety of Life at Sea (SOLAS),” International Maritime Organization, London, UK. [Online]. Available: http://www.imo.org/About/Conventions/ListOfConventions/Pages/International-Convention-for-the-Safety-of-Life-at-Sea-(SOLAS),-1974.aspx

[23] N. Bowditch, *The American Practical Navigator*, National Imagery and Mapping Agency, Paradise Cay, Oct. 01, 2002.

[24] W. Stutzman and G. Theile, *Antenna Theory and Design*, Danvers, MA: John Wiley & Sons, Dec. 29, 1997.

[25] M. E. Celebi, “Effective initialization of k-means for color quantization,” 16th IEEE International Conference on Image Processing (ICIP), pp. 1649-1652 Nov. 7–10, 2009.

[26] H. Evers and S. Gerky, “The strategic importance of the Straits of Malacca for world trade and regional development,” Center for Development Research, University of Bonn, Bonn, Germany, Working Paper Nr. 17, 2006.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California